

June 1989

Report No. STAN-CS-89-1263
Also Numbered KSL-89-52
Thesis

2

THIS IS A COPY

AD-A219 003

HYPOTHESIS FORMATION AND QUALITATIVE REASONING IN MOLECULAR BIOLOGY

by

Peter Dornin Karp

Department of Computer Science

Stanford University
Stanford, California 94305



DTIC
ELECTE
MAR 13 1990
S B D

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

90 03 12 091

DARPA INSERT SHEET

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188
Exp Date Jun 30, 1986

1a REPORT SECURITY CLASSIFICATION unclassified		1b RESTRICTIVE MARKINGS	
2a SECURITY CLASSIFICATION AUTHORITY		3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release: Distribution Unlimited	
2b DECLASSIFICATION/DOWNGRADING SCHEDULE			
4 PERFORMING ORGANIZATION REPORT NUMBER(S) STAN-CS-89-1263		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a NAME OF PERFORMING ORGANIZATION Computer Science Department	6b OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION	
6c. ADDRESS (City, State, and ZIP Code) Stanford University Stanford, CA 94305		7b. ADDRESS (City, State, and ZIP Code)	
8a NAME OF FUNDING, SPONSORING ORGANIZATION DARPA	8b OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00039-86C-0033	
8c. ADDRESS (City, State, and ZIP Code) 1400 Wilson Blvd. Arlington, VA 22209		10. SOURCE OF FUNDING NUMBERS	
		PROGRAM ELEMENT NO.	PROJECT NO.
		TASK NO.	WORK UNIT ACCESSION NO.
11 TITLE (Include Security Classification) HYPOTHESIS FORMATION AND QUALITATIVE REASONING IN MOLECULAR BIOLOGY			
12 PERSONAL AUTHOR(S) PETER D. KARP			
13a TYPE OF REPORT thesis	13b TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Year, Month, Day) 1989 June	15. PAGE COUNT 337
16 SUPPLEMENTARY NOTATION			

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This dissertation investigates scientific reasoning from a computational perspective. The investigation focuses on a program of research in molecular biology that culminated in the discovery of a new mechanism of gene regulation in bacteria, called attenuation. The dissertation concentrates on a particular type of reasoning called hypothesis formation. Hypothesis-formation problems occur when the outcome of an experiment predicted by a scientific theory does not match that observed by a scientist. I present methods for solving hypothesis-formation problems that have been implemented in a computer program called HYPGENE. This work is also concerned with how to represent theories of molecular biology in a computer, and with how to use such theories to predict experimental outcomes; I present a framework for performing these tasks that is implemented in a program called GENSIM. I tested both HYPGENE and GENSIM on sample problems that biologists solved during their research on attenuation. The dissertation includes a historical study of the attenuation research. This study is novel because it examines a large, complex, and modern program of scientific research. (over)

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION	
22a NAME OF RESPONSIBLE INDIVIDUAL Edward Feigenbaum		22b. TELEPHONE (Include Area Code)	22c OFFICE SYMBOL

Abstract (cont'd)

The thesis treats hypothesis formation as a design problem, and uses design methods to solve hypothesis-formation problems. The HYPGENE program reasons backward from an error in a GENSIM prediction of an experimental outcome. It uses hypothesis-design operators to design modifications to the initial conditions of the experiment, and to the biological theory, such that the new predicted outcome of the experiment computed by GENSIM will match the observation. This approach is largely domain-independent because most design operators include no domain concepts. The design operators are complete in that they can synthesize any hypothesis that can be represented within the GENSIM framework. HYPGENE uses a planner to satisfy its design goals. This method of hypothesis formation is efficient because it is goal directed (in contrast to previous generate-and-test approaches), and because reference experiments can be used both to guide the generation of hypotheses, and to filter hypotheses. A reference experiment has initial conditions that are similar to those of the anomalous experiment, but has an outcome that is correctly predicted by the theory.

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

HYPOTHESIS FORMATION AND QUALITATIVE REASONING
IN
MOLECULAR BIOLOGY

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Peter Domain Karp

June, 1989

© Copyright 1989 by Peter Dornin Karp

All Rights Reserved

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Edward A. Feigenbaum
(Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Bruce G. Buchanan
(Department of Computer Science
University of Pittsburgh)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Peter E. Friedland
(NASA Ames Research Center)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Charles Yanofsky
(Department of Biology)

Approved for the University Committee on Graduate Studies:

Dean of Graduate Studies

Abstract

This dissertation investigates scientific reasoning from a computational perspective. The investigation focuses on a program of research in molecular biology that culminated in the discovery of a new mechanism of gene regulation in bacteria, called *attenuation*. The dissertation concentrates on a particular type of reasoning called *hypothesis formation*. Hypothesis-formation problems occur when the outcome of an experiment predicted by a scientific theory does not match that observed by a scientist. I present methods for solving hypothesis-formation problems that have been implemented in a computer program called HYPGENE. This work is also concerned with how to represent theories of molecular biology in a computer, and with how to use such theories to predict experimental outcomes; I present a framework for performing these tasks that is implemented in a program called GENSIM. I tested both HYPGENE and GENSIM on sample problems that biologists solved during their research on attenuation. The dissertation includes a historical study of the attenuation research. This study is novel because it examines a large, complex, and modern program of scientific research.

The ~~thesis~~ treats hypothesis formation as a *design* problem, and uses design methods to solve hypothesis-formation problems. The HYPGENE program reasons backward from an error in a GENSIM prediction of an experimental outcome. It uses hypothesis-design operators to design modifications to the initial conditions of the experiment, and to the biological theory, such that the new predicted outcome of the experiment computed by GENSIM will match the observation. This approach is largely domain-independent because most design operators

include no domain concepts. The design operators are complete in that they can synthesize any hypothesis that can be represented within the GENSIM framework. HYPGENE uses a planner to satisfy its design goals. This method of hypothesis formation is efficient because it is goal directed (in contrast to previous generate-and-test approaches), and because *reference experiments* can be used both to guide the generation of hypotheses, and to filter hypotheses. A reference experiment has initial conditions that are similar to those of the anomalous experiment, but has an outcome that is correctly predicted by the theory.

Acknowledgements

Many graduate students work closely with only one advisor. I had the benefit of working fairly closely with all of my reading-committee members. I am indebted to all the members of my committee for their encouragement and support — both professional and personal — during my years at Stanford.

Bruce Buchanan's knowledge of science — both as observer and practitioner — have been constant sources of *inspiration to me*. Bruce had a *very strong influence on my work* through our many stimulating meetings.

I thank Ed Feigenbaum for always pushing me to do better and more ambitious work. Ed has taught me how important it is for a researcher to set difficult goals for his or her work.

I probably worked most closely with Peter Friedland, particularly on the research presented in Chapter 4. While constructing the history of attenuation, Peter taught me the art of knowledge engineering. Peter and I wrote much of Chapter 4 and the first part of Chapter 3 together; this type of collaboration is in my opinion one of the most important learning experiences for a graduate student.

Charles Yanofsky was the perfect collaborator for the interdisciplinary research project I tackled: He never seemed to tire of the biology questions with which we nonbiologists barraged him. His memory of his past experiments, hypotheses, and states of knowledge is extraordinary.

Much of the research for the historical study contained in Chapter 4 was the joint work of Peter Friedland, René Bach, and myself. We spent many enjoyable hours together trying to

understand the mysteries of attenuation. I am indebted to Paul Rosenbloom for suggesting that we write the historical study in the first place. I also thank René for the many hours he spent teaching me INTERLISP. The biologist Robert Landick collaborated with me on several aspects of the historical study and on my models of the trp operon.

A number of people contributed helpful comments on different drafts of various chapters of the dissertation, including Devika Subramanian, Jeff Shrager, Liam Peyton, Robert Engelmores, Richard Washington, Lindley Darden, Jonathan Amsterdam, Stuart Russell, and Richard Keller.

Lyn Dupre edited this dissertation and improved its form tremendously. In so doing she indirectly edited my brain, thus improving my writing skills substantially.

My fellow students at Stanford provided many stimulating seminar hours and discussions; thanks in particular to Andy Golding, Benjamin Grosz (particularly for discussions about the ATMS), Stuart Russell, Devika Subramanian, David Wilkins, and Eric Horvitz. Max Hailperin helped me understand the ways of LaTeX.

Tom Dietterich kindly provided an ATMS implementation that I used for a number of experiments. IntelliCorp generously provided Stanford with its KEE knowledge-representation tool. All my work relied heavily on KEE's representation facilities.

The Symbolic Systems Research Group has provided an unparalleled computing environment for my research. General thanks go to Tom Rindfleisch, Bill Yeager, and Rich Acuff. Special thanks to Christophers Schmidt and Lane, the D-machine wizards who never ceased to amaze me with their knowledge and powers. Many times I was sure that I was nailed by a particular obstacle in the implementation, only to have them solve it. Special thanks also to Bob Tucker.

My friends made my years at Stanford immensely enjoyable. Thanks to Hearse, Cohnan (Reidan?), Rambo, Anne, Dan L., Kate, Eric, Meems, Alan, Suze, Barb, Roy, Stu, Dan S., Don, Diann, the Sweat Buddies, the Brewmates, and the Burger Club.

My family was always there with support, encouragement, and fun times. My regret is that Stanford is not closer to Philadelphia.

This work was supported by funding from NSF grant MCS83-10236, NIH grant RR-00785, and DARPA contract N00039-83-C-0136.

Contents

Abstract	iv
Acknowledgements	vi
1 Introduction	1
1.1 Scientific Reasoning Tasks	2
1.2 Previous Work on Hypothesis Formation	6
1.2.1 Classes of Domain Problems	6
1.2.2 Complexity of Domain Theories	7
1.2.3 Hypothesis-Formation Methods	8
1.3 Statement of the Thesis	9
1.4 Overview of the Dissertation	11
1.4.1 A Historical Study of Attenuation	11
1.4.2 Theories of the Trp Operon	13
1.4.3 Hypothesis Formation by Design	15
1.4.4 Results	18
1.5 Readers Guide	18
1.6 The Evaluation of Scientific Hypothesis-Formation Programs	19
2 Background Knowledge of Molecular Biology	22

2.1	Molecular Genetics Background	22
2.1.1	Proteins and the Genetic Material	23
2.1.2	Transcription and Translation	25
2.1.3	Mutations	27
2.1.4	Regulation	28
2.2	Techniques for Research on the trp Operon	31
2.2.1	Observational Techniques	32
2.2.2	Manipulative Techniques	34
2.2.3	An Example Experiment	36
2.2.4	The Manipulation Versus Observation Distinction, Reconsidered	37
2.3	Summary	37
3	Declarative Device Models of the Trp Operon	38
3.1	Scientific Theories	40
3.2	Terminology	41
3.3	Model 1: The Rule Model	43
3.4	Model 2: The Fixed State-Variable Model	45
3.4.1	Techniques	46
3.4.2	An Example Model 2 Prediction	62
3.4.3	Analysis of Model 2	66
3.4.4	Summary of Model 2	67
3.5	Model 3: The Process Model (GENSIM)	68
3.5.1	An Example	69
3.5.2	Class Knowledge Base	69
3.5.3	Process Knowledge Base	73
3.5.4	Representational Tradeoffs	77

3.5.5	An Assessment of GENSIM's Knowledge	78
3.5.6	The Process Interpreter	79
3.5.7	GENSIM Results	92
3.5.8	Conclusions	92
4	A Reconstruction of the Discovery of Attenuation	96
4.1	Study Methodology	98
4.2	On Scientific Theories	99
4.3	Organization of Yanofsky's Laboratory	100
4.4	Annotated Chronology of the Research	102
4.4.1	Research on the Trp Operon: Synopsis	110
4.4.2	State 1: Exploration Within an Articulated Theory	115
4.4.3	State 2: A Concrete Anomaly and Preliminary Discrimination of Its Causes	121
4.4.4	State 3: Confirmation of the Theory of Premature Transcription Termination	125
4.4.5	State 4: A Mechanism for Attenuation — Role of tRNA ^{trp} and Secondary Structure	130
4.4.6	State 5: The Attenuation Mechanism Is Confirmed and Refined	137
4.5	Analysis	141
4.5.1	Modes of Scientific Exploration	142
4.5.2	Theory Generation	154
4.5.3	The Histidine Operon	159
4.6	Summary	160
5	Hypothesis Formation by Design	163
5.1	The Hypothesis-Formation Problem	164
5.2	A Design System for Hypothesis Generation	170

5.3	An Example Hypothesis-Formation Problem	175
5.4	The Design Goals	181
5.4.1	Goals Involving Assertions	182
5.4.2	Quantitative Design-Goals	182
5.5	Design Operators	183
5.5.1	Initial Condition Design Operators	185
5.5.2	Quantity-Hypothesis Design Operators	188
5.5.3	Possible Process-Design Operators	192
5.5.4	Possible Class-KB Design Operators	194
5.6	Detection of Valid Hypotheses	196
5.7	The Computational Complexity of HYPGENE	198
5.8	Control of HYPGENE's Search	199
5.8.1	Simplicity	201
5.8.2	Domain Knowledge	202
5.8.3	Operator Precedence	203
5.8.4	Reference Experiments	204
5.9	Summary	215
6	Hypothesis Formation Details	218
6.1	The Implementation of HYPGENE	218
6.1.1	HYPGENE's Search	219
6.1.2	Representation of Design Goals	220
6.1.3	Expansion of Search States	222
6.1.4	Execution of Design Operators	224
6.1.5	Goal Satisfaction	225
6.1.6	Detection of Loops	226

6.1.7	Violation of Previously Satisfied Goals	226
6.2	Literature Survey	227
6.2.1	Dietterich's PRE	227
6.2.2	Simmons' GORDIUS	230
6.2.3	Rajamoney's COAST	232
6.2.4	Wilkins' ODYSSEUS	234
6.2.5	Lenat's AM and EURISKO	236
6.2.6	The Early Chemists	237
6.2.7	DENDRAL	240
6.2.8	Meta-DENDRAL	242
6.2.9	Koton's GENEX	243
6.2.10	Kulkarni's KEKADA	244
6.2.11	Tong's DONTÉ	245
6.2.12	Other Work	245
6.3	Lessons Learned from GENSIM and HYPGENE	246
6.3.1	On Writing Process Preconditions	247
6.3.2	Process Granularity	248
6.3.3	Representational Tradeoffs in Process Preconditions	249
6.3.4	Predicate Calculus and Frames	250
6.4	Summary	252
7	Experiments	253
7.1	GENSIM Trials	254
7.1.1	The Trp Biosynthetic Pathway	254
7.1.2	The Trp Biosynthetic Pathway with a Mutant Enzyme	256
7.1.3	Transcription of the Trp Leader Region	256

7.1.4	The Full Trp System	257
7.2	HYPGENE Trials	259
7.2.1	Trp Biosynthetic-Pathway Hypotheses	261
7.2.2	Transcription-Initiation Hypotheses	264
7.2.3	Attenuation Hypotheses	267
7.3	Summary	273
8	Conclusions	275
8.1	The Historical Study of Attenuation	275
8.1.1	Limitations	275
8.1.2	Contributions	276
8.2	Declarative Device Modeling	276
8.2.1	Limitations	277
8.2.2	Contributions	277
8.3	Hypothesis Formation	278
8.3.1	Limitations	279
8.3.2	Contributions	280
8.4	Future Work	282
A	Process Knowledge Base	284
B	Class Knowledge Base	296
C	Attenuation Paper Summaries	300
C.1	State 1	300
C.2	State 2	302
C.3	State 3	302
C.4	State 4	304

C.5 State 5	305
-----------------------	-----

Bibliography	307
---------------------	------------

List of Figures

1.1	Scientific reasoning tasks	3
1.2	Complexity of hypothesis-formation problem domain theories	8
1.3	The relationship between the GENSIM and HYPGENE programs	16
2.1	Overview of the regulation of the tryptophan operon	24
2.2	The genetic code	26
2.3	Example of protein synthesis	27
2.4	Trp operon structural genes and the proteins they code for	29
3.1	Fixed state-variable network describing the trp operon	48
3.2	Generalization hierarchy of mathematical functions	52
3.3	A sample mapping	54
3.4	A prediction generated by model 2: tick 1	64
3.5	A prediction generated by model 2: tick 2	65
3.6	The objects in a transcription experiment	71
3.7	A simple reaction	81
3.8	Using an ATMS to share object structures	91
3.9	Simulation of the trp system using model 3: part 1	93
3.10	Simulation of the trp system using model 3: part 2	94
4.1	The major states of knowledge possessed by the trp system researchers.	103

4.2	Assignments of the trp literature to the states of knowledge	105
4.3	The trp literature organized by time and subject matter	106
4.4	Evolution of the theory of the regulation of the trp operon	107
4.5	Alternative hypotheses generated by the trp researchers: part 1	108
4.6	Alternative hypotheses generated by the trp researchers: part 2	109
4.7	Alternative secondary structures in the trp operon leader region	114
4.8	Modes of scientific exploration	143
5.1	The error in a GENSIM prediction	169
5.2	HYPGENE's execution cycle	174
5.3	A sample GENSIM prediction: repression	176
5.4	HYPGENE creates a mutation object	179
5.5	HYPGENE determines the specificity of the mutation	180
5.6	HYPGENE inserts the mutation into a binding site	181
5.7	Initial-condition design operators and process-design operators	184
5.8	Quantity-hypothesis design operators	189
5.9	A sample reaction network	190
5.10	A sample reaction network	197
5.11	A hypothetical reference experiment	206
5.12	Reference experiment examples: cases 1-4	209
5.13	Reference Experiment examples: cases 5-8	210
7.1	Trp biosynthetic-pathway experiment: initial conditions	254
7.2	Trp biosynthetic-pathway experiment: predicted outcome	255
7.3	Trp biosynthetic-pathway experiment: object structures	255
7.4	Leader-region-transcription experiment: initial conditions	257
7.5	Leader-region-transcription experiment: predicted outcome	257

7.6	Internal structure of a transcription-elongation complex	257
7.7	Leader-region-transcription experiment: object structures	258
7.8	Full trp system experiment: initial conditions	260
7.9	Trp biosynthetic-pathway experiment: hypotheses	262
7.10	Trp biosynthetic-pathway experiment: quantitative hypotheses	263
7.11	Transcription-initiation experiment hypotheses	265
7.12	Transcription-initiation experiment: hypotheses	266
7.13	Jackson-Yanofsky experiment: predicted outcome	268
7.14	Jackson-Yanofsky experiment: hypotheses	269
7.15	Jackson-Yanofsky experiment: hypothesis 4	270
A.1	The process knowledge base	285
B.1	The top third of the class KB.	297
B.2	The middle third of the class KB.	298
B.3	The bottom third of the class KB.	299

Chapter 1

Introduction

This dissertation investigates scientific reasoning from a computational perspective. The investigation focuses on a program of research in molecular biology that culminated in the discovery of a new mechanism of gene regulation in bacteria. The dissertation concentrates on a particular type of reasoning called *hypothesis formation*. Hypothesis-formation problems occur when the outcome of an experiment predicted by a scientific theory does not match that observed by a scientist. This work is also concerned with how to represent theories of molecular biology in a computer, and how to use such theories to predict experiment outcomes.

I present a framework for representing theories of molecular biology, and for predicting experimental outcomes, which is embodied in a program called GENSIM. I also present methods for solving hypothesis-formation problems. These methods have been implemented in a computer program called HYPGENE. Both HYPGENE and GENSIM have been tested on sample problems that biologists solved during their research on the gene-regulation mechanism called *attenuation*. I performed a historical study of this biological research, which is included in the dissertation.

This introductory chapter puts the thesis work in perspective. Section 1.1 identifies a large set of reasoning tasks that scientists perform, and indicates which tasks are addressed in this

dissertation and which are not. Section 1.2 reviews past artificial intelligence (AI) research in hypothesis formation to assess the state of the art. Section 1.3 contains a statement of the thesis, and Section 1.4 gives a detailed overview of the methods and techniques presented here to show how this work advances the state of the art. Section 1.5 is a readers guide to the dissertation. Section 1.6 considers the question of how to evaluate machine-learning programs.

1.1 Scientific Reasoning Tasks

I view science as a goal-oriented activity the objective of which is to improve the predictive performance of scientific theories.¹ In pursuit of this objective, scientists solve a number of different reasoning problems, such as designing experiments and formulating hypotheses. This dissertation is concerned with some, but not all, of these problems. In this section I present a brief overview of the reasoning tasks that scientists perform, to establish a broad perspective on scientific activity.

Figure 1.1 lists a number of scientific reasoning tasks, and shows how the tasks might be arranged in a flow chart of scientific activity. The organization in Figure 1.1 is only one of many possible organizations that could apply to different scientific disciplines in different stages of maturity.

Let us consider an example of how the reasoning tasks in Figure 1.1 would be used by a scientist who wished to test an existing theory of gene regulation to determine whether that theory explained the regulation of a newly discovered set of genes. Already, the scientist has selected an experimental goal based on personal, scientific, and social preferences about what problems are interesting and worth solving.

To achieve that goal, he must design an experiment involving the regulation of the genes, using general knowledge of experimental techniques (which is hidden in "Theory" in Figure 1.1),

¹This view of science is simplified, but it will serve our purposes.

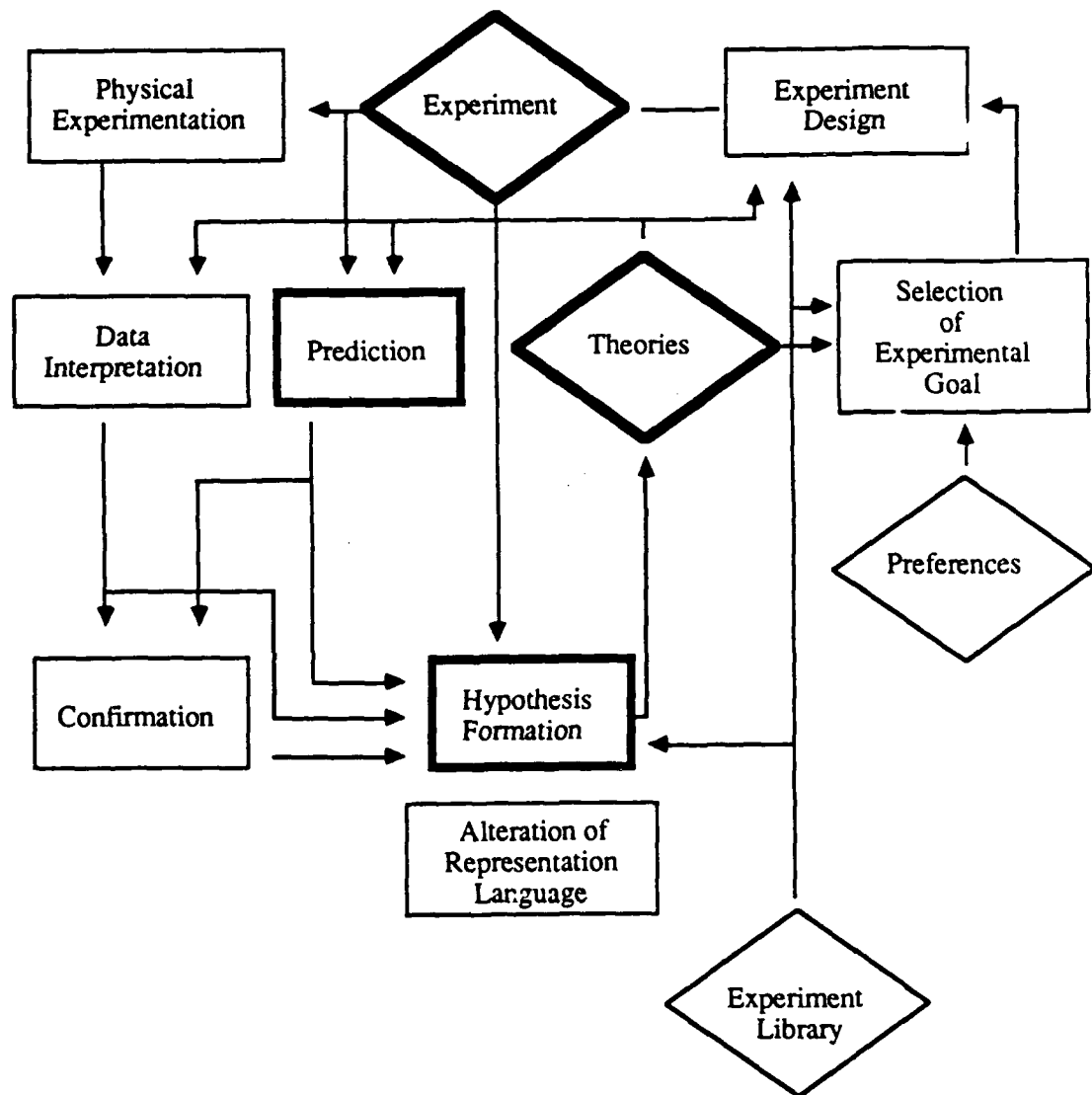


Figure 1.1: General scientific reasoning tasks and their relationships to one another.

and specific knowledge of past experiments that he has performed.² The experiment-design process should generally yield an experiment that tests the theory quickly, inexpensively, and unambiguously, subject to the constraint that it is not technically possible to measure some aspects of gene-regulation systems. He might design an experiment that examines the effects of different mutations on the regulation of the genes.

When the scientist performs this experiment in his laboratory, he obtains various physical measurements from his laboratory instruments. He must interpret these raw data to infer, for example, what the color of a solution in a test tube implies about the degree to which the genes were turned on by the cell. In addition to observing the outcome of the experiment in the laboratory, the scientist must use his theory to predict the outcome of the experiment. The theory should predict what chemical reactions occur in this experiment, and how the rates of the reactions affect the rate at which the genes are expressed. The scientist must compare this predicted rate of gene expression against the rate he observes in the laboratory. This comparison must include some tolerance to account for the accuracy with which predictions and observations can be made.

If the predicted and observed outcomes of the experiment do not match within the established tolerance, this situation may be an opportunity to improve the predictive performance of the theory — or it may signal a problem with the experiment itself. This mismatch between prediction and observation calls for hypothesis formation. The scientist must alter his theory or his beliefs about the laboratory experiment to eliminate the discrepancy between the two. The scientist might alter his beliefs about the experiment for a variety of reasons, such as if he decided: that impurities were present in the experiment, that his experimental procedures were in error, that his laboratory instruments were miscalibrated, or that his understanding of what his laboratory instruments were measuring was flawed. If he trusted his experimental results, however, he would alter his theory. The new theory would have to cover the outcome

²The MOLGEN work of Friedland and of Stefik addressed the experiment-planning problem; my work evolved from theirs, but addresses different problems [Friedland79,Stefik80].

of the current experiment, and those of any previous experiments that had been performed on similar gene-regulation systems.

Note that to perform these reasoning tasks, the scientist must possess knowledge structures that describe theories, single experiments, and libraries of experiments.

In this dissertation I address only some of the preceding tasks. I am concerned with how to represent theories and experiments, and with how to use theories to predict outcomes of experiments. How can we represent the theory of gene regulation employed by the scientist in this example, and how can we use such a theory to predict the outcome of a gene-regulation experiment? I also address the problem of hypothesis formation. How does a scientist determine which of his beliefs about the experiment could be wrong? How does he decide how to alter his theory of gene regulation such that its predictions are consistent with experimental outcomes? These are the two main foci of my work, although I have also considered how scientists select experimental goals.

A dissertation that addresses these issues is interesting for several reasons. First, it will improve our *understanding* of scientific-reasoning processes. Second, if we understand these reasoning tasks better, we will be able to build computational tools to aid scientists in performing these tasks. For example, a scientist who is formulating hypotheses to explain an unexpected experimental outcome might use a computer program to aid him in formulating a thorough set of alternative hypotheses, or in determining whether a candidate hypothesis is consistent with data from previous experiments that are recorded within the computer. Humanity has accumulated so much scientific knowledge, and the scientific problems that we are now tackling are becoming so complex, that computer tools may soon become indispensable to scientists.

1.2 Previous Work on Hypothesis Formation

Here we consider past artificial-intelligence research on the problem of scientific-hypothesis formation to assess the state of the art.³ This section compares previous work in this area along several dimensions: the classes of domain problems that previous researchers have addressed, the complexity of both the domain theories they have used and the domain problems they have solved, and the hypothesis-formation methods they have developed. This section is not particularly detailed — see Chapter 6 for in-depth comparisons of problem-solving methods. Although I include my work in the tables that summarize this discussion, I wait until Section 1.4 to describe how my work advances the state of the art.

1.2.1 Classes of Domain Problems

Different researchers have used different methodologies to study hypothesis formation. Some researchers have studied historical examples of scientific hypothesis-formation problems and have constructed programs that reproduce, with varying degrees of accuracy, the reasoning by which scientists solved those problems. Other researchers have built programs that aid scientists in answering unsolved scientific questions. A third group of researchers has explored hypothesis formation in synthetic scientific domains that do not accurately include either past or present scientific problems. Finally, a fourth group of researchers has studied the formation of hypotheses for problems that are not scientific in nature, but that bear strong similarities to scientific hypothesis-formation problems. Table 1.1 summarizes these different classes of domain problems. Note that in this section, we are concerned with the *type* of a problem domain, not with the *complexity* of a domain. The programs in Table 1.1 are described in the following publications: BACON, GLAUBER, STAHL, DALTON [Langley87]; KEKADA [Kulkarni88]; STAHLp [RoseL86]; DENDRAL [Lindsay80]; Meta-DENDRAL [BuchananM78]; RX

³Other researchers have used a variety of terms for what I call hypothesis formation, including: *theory formation*, *theory revision*, *theory debugging*, *interpretation*, *discovery*. *Hypothesis formation* subsumes all these terms; for example, some hypotheses are revisions to theories.

Hypothesis-Formation Problem Types			
Historical	Unsolved	Synthetic	Other
BACON	DENDRAL	COAST	PRE
GLAUBER	Meta-DENDRAL	GORDIUS	
STAHL	RX		
STAHLp	Soo's System		
DALTON	GORDIUS		
KEKADA	PROTEAN		
Darden's System	ARIADNE		
PI			
HYPGENE			

Table 1.1: A classification of the types of problems that previous AI hypothesis-formation programs have solved.

[Blum82d]; GORDIUS [Simmons88]; COAST [Rajamoney88]; PRE [Dietterich84]; PROTEAN [Altman89,Duncan89]; ARIADNE [Lathrop87]; Darden and Rada's system [DardenR88]; Soo's system [Soo87]; PI [ThagardH85].

1.2.2 Complexity of Domain Theories

Real scientific problems usually involve large, complicated theories and experiments. We can characterize this complexity in terms of the number of classes of different objects in the domain, the number of objects in typical experiments in the domain, the complexity of individual objects (such as the number of object attributes and the size of object part-whole structures), the number of rules that constitute the domain theory, and the internal complexity of individual rules (number of clauses in antecedent and consequent; use of negation, disjunction, and quantification within clauses; use of recursion). Another measure of complexity is whether a theory describes real-valued state variables of a physical system, and, if it does, whether it represents the variables quantitatively or using a qualitative representation [Bobrow84].

Figure 1.2 provides an approximate ranking of the complexities of the domain theories used

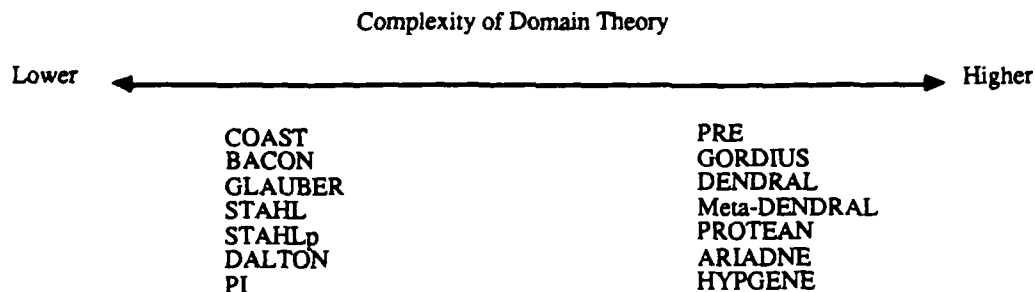


Figure 1.2: The complexity of domain theories employed by previous hypothesis-formation programs.

in different hypothesis-formation programs, using the preceding measures of complexity.

1.2.3 Hypothesis-Formation Methods

Previous researchers have developed three different classes of methods to solve hypothesis-formation problems, as shown in Table 1.2. Broadly speaking, these methods are variations of generate-and-test (the variations include DENDRAL's planning phase, and forms of heuristic guidance), probabilistic updating and statistical analysis, and methods that have been called by several names including *explanation-based*, *causal*, and *constraint propagation*. One difference is that the programs in the first column generate hypotheses from naught, whereas the programs in the third column are given hypotheses that are almost correct, and the programs revise these hypotheses to make them correct. For example, the GLAUBER program formulates chemical reaction laws that are consistent with a set of chemical reactions, but GLAUBER does not start with a theory of chemistry. Simmons' theory debugger, on the other hand, corrects a buggy interpretation of how a geologic region was formed.⁴ The programs in the first column use syntactic operators to generate hypotheses, then test the correctness of the hypotheses. The programs in the third column analyze incorrect predictions made by the theory with which they start to determine how to modify the theory such that its predictions are correct. They scrutinize a dependency trace or explanation structure that records dependencies among the

⁴Simmons' work actually falls under both columns 1 and 3; his system first generates a hypothesis, and then removes any bugs that the hypothesis contains.

Generate-and-Test	Hypothesis-Formation Methods	
	Probabilistic	Explanation/Causal/Constraint-Based
BACON	PROTEAN	COAST
GLAUBER	RX	PRE
STAHL		GORDIUS
DALTON		STAHLp
KEKADA		HYPGENE
DENDRAL		
Meta-DENDRAL		
Soo's System		
ARIADNE		

Table 1.2: A classification of the methods that previous AI hypothesis-formation programs have employed.

prediction, the initial conditions of the experiment, and the theory. The approach in column 3 is generally more efficient because it is more goal directed, and because it starts with more information (an almost-correct theory). I also believe that the approach in column 3 is a more realistic view of science — that theories rarely arise in a vacuum from data alone, but usually are created in the context of existing theoretical knowledge.

1.3 Statement of the Thesis

This dissertation presents methods for solving two different problems: (1) the problem of representing scientific theories within a computer and using these theories to predict outcomes of experiments, and (2) the problem of formulating hypotheses to account for errors in the predictions of a theory. The methods were developed and tested within the domain of molecular biology, so my claims about their usefulness must be limited to this domain. (Chapter 8 presents reasons for thinking that these techniques will be applicable to other domains, and considers what properties other domains must have for the techniques to work there.)

This document advances the following thesis:

- The methods in Chapter 3 are sufficient for representing qualitative scientific knowledge about objects and processes in molecular biology and biochemistry, such that the knowledge can be used to predict experimental outcomes, and such that other programs can reason about and modify this knowledge.
- We profit from treating hypothesis formation as a *design problem*. Further, the design methods presented in Chapter 5 provide both a flexible and an efficient framework for solving hypothesis-formation problems. The framework is flexible because it is syntactically complete, is largely domain-independent, and includes a planner that can reason about complex domain processes.⁵ The framework is efficient because the problem solver's search is focused on prediction errors, and because reference experiments can be used both to evaluate hypotheses, and to guide the process of hypothesis generation.

This view of hypothesis formation as design is important both conceptually and pragmatically. Conceptually, design provides a new framework for thinking about what scientists do. Design becomes a metaphor for scientific activity in which we view scientists as architects of complex structures that happen to exist only in minds; namely, scientific theories. Within the design paradigm, we view theory formation as a goal-directed endeavor. This view implies that scientific theories are artificial entities, which conflicts with Simon's view that scientific theories are natural entities [Simon69]. At this conceptual level, we can think about science within a number of other frameworks as well; we can think of theory formation as diagnosis, or as debugging, or within the performance-oriented framework of machine learning [Simon82,DietterichB81]. Note that the design paradigm does not just apply to scientific hypothesis formation; we can also think of both diagnosis and debugging as design problems.

Pragmatically, words such as *design*, *diagnosis*, and *debugging* are too vague to be particularly interesting; computer scientists seek concrete techniques that will solve real problems. AI

⁵The framework is syntactically complete because it can generate all theories that can be represented using the GENSIM representation language.

researchers have developed a number of concrete techniques for solving particular problems in design, diagnosis, and debugging [Tong88,Davis84,Genesereth84,deKleerW86,Sussman73]. In this dissertation I show how techniques that have been developed to solve design and planning problems can be used to solve hypothesis-formation problems. The pragmatic value of the design metaphor is that it suggests a number of existing ideas that we can apply to the problem of hypothesis formation. I have explored the use of only some design techniques to formulate hypothesis; among the techniques I have not explored are designing a theory at different levels of abstraction, maintaining a design history for a theory to facilitate future theory revisions, maintaining a library of theories so that we may create new theories by redesigning old ones, and planning about the theory-design process itself. So the design framework provides us with the particular techniques that this dissertation shows can be used to solve hypothesis-formation problems, and in addition it provides us with the potential use of several other techniques.

1.4 Overview of the Dissertation

This section provides an overview of the results of the thesis research. It puts the methods and techniques presented here in perspective by referring to the summary of previous hypothesis-formation research discussed in Section 1.2.

1.4.1 A Historical Study of Attenuation

In this dissertation I use the historical approach shown in Table 1.1 to study scientific reasoning. That is, I measure the success of my methods by their ability to solve reasoning problems that scientists have faced in the past. I studied a program of research in molecular biology in which Professor Charles Yanofsky and other scientists discovered a novel mechanism of gene regulation in bacteria, called *attenuation*. The biologists studied a set of genes called the *tryptophan operon*, or *trp operon*.

One of the most important contributions of this work is that I studied more complex

and realistic hypothesis-formation problems than have previous researchers. The attenuation research is more complex than any of the problems studied by researchers in the Historical, Synthetic, or Other columns of Table 1.1 (attenuation research consumed over 50 person-years of effort). Complexity is important not only because it ensures that we address realistic problems, but also because it allows us to identify the context within which a research problem arose. Also important is that this biological research was performed very recently (in the 1960s and 1970s), so we can be more certain of its accuracy than of the accuracy of our knowledge of the events surrounding scientific discoveries made hundreds of years ago. Consider the BACON program's derivation of Kepler's laws of planetary motion. How certain are we that Kepler sat down at his desk with a particular array of numbers, and that he immediately derived his laws from them? How much do we know about the context in which that problem arose that would tell us why he sat down with those particular numbers, why he chose to measure those particular variables in the first place, or what experimental apparatus he employed? Finally, it is important to question the realism of the problems addressed under the Synthetic and Other categories in Table 1.1; are they sufficiently similar to real scientific problems? For example, if search control is a central problem in hypothesis formation, and a program in a synthetic domain performs essentially no search, then it has missed a crucial set of issues in this research area.

Chapter 4 contains an in-depth study of the process by which Dr. Yanofsky and his colleagues discovered attenuation. It is based on information obtained from the scientific publications that these biologists authored, and from interviews that I conducted with the biologists. In the first phase of the analysis, I produced a conceptual reconstruction of what knowledge the biologists possessed about the *trp* operon at different times. This reconstruction also describes the experiments that the biologists performed, and the alternative hypotheses that the biologists proposed to explain the outcomes of their experiments. Thus, I identified sample

hypothesis-formation problems, experiment-prediction problems, and experiment-design problems that can be used to test computational methods for performing these tasks. Some of these examples are used in this dissertation; others are topics for future research.

In the next phase of the analysis, I searched for patterns in the differences between successive states of the biologists' knowledge. These differences existed because the biologists altered their theories of the *trp* operon. I searched for patterns in the differences that I assumed would be due to the reasoning methods that the scientists used to derive new theories from old. This analysis suggests that biologists use *theory-modification operators* to modify a theory and thus to alter its predictions. My analysis also supports the conjecture that scientists use four different *modes of scientific exploration* to determine what types of experiments to perform next from a given state of research. A mode of exploration selects experiments based on the number of theories entertained at a given moment, and/or these theories' relative credibilities.

1.4.2 Theories of the Trp Operon

The complexity of the theories and experiments in the *trp*-operon gene-regulation system is equal to or exceeds the domain theories that previous AI researchers have used to study hypothesis formation (see Figure 1.2). My work involves a domain theory that is significantly more complex than is the classic work in "scientific discovery" (such as the BACON program), but that is comparable in complexity to the theories used in the programs on the right side of Figure 1.2. I developed new methods for representing theories of molecular biology that are based on current AI research in qualitative reasoning about physical systems. These methods are implemented in three different models of the *trp* operon, which are described in Chapter 3. I will ignore the first model here because it is the least interesting.

The second model focuses on quantitative aspects of the *trp* operon, such as the degree to which the genes within the operon are turned on, and the rates of the chemical reactions that control the operon. This model links different state variables of the operon to form a constraint

network, and predicts the outcomes of experiments by propagating variable values through the network. Biologists often have only partial, qualitative knowledge of these values and dependencies. For example, the precision with which they know values of state variables can lie in between purely quantitative precision, and the gross qualitative precision that researchers in qualitative physics have explored [deKleer84]. Model 2 explores new representations both for the values of state variables and for the mathematical dependencies between variables. It also presents inference mechanisms for computing predictions using these representations.

The third model (GENSIM) focuses on representing the attributes and structures of the objects that make up the *trp* operon, and on simulating the chemical reactions that occur in different experiments. Model 2 is relevant to a relatively small number of experiments because its fixed state-variable network makes many assumptions about what objects are present in an experiment. Model 3 allows the user to specify what objects are present at the start of an experiment; the GENSIM simulator then predicts what additional objects are present at the end of the experiment. Model 3 provides a framework that includes several parts: A *class knowledge base* defines a taxonomic hierarchy of the classes of biological objects in bacteria that are related to the regulation of the *trp* operon. A *process knowledge base* describes the chemical reactions that can occur among biological objects. Users describe experiments in a *simulation knowledge base* by creating the particular objects (instantiated from the known classes of objects) that are present in the experiment.

The GENSIM simulator uses information in the process knowledge base to determine what reactions occur among the objects in an experiment; reactions create new objects, which can cause additional reactions. GENSIM defines a *qualitative chemistry* — a framework for reasoning about chemical reactions. Chapter 3 explores a number of issues in qualitative simulation. How can we represent and instantiate part-whole structures for classes of objects? How should we manage the objects that are created during a simulation — should chemical reactions modify existing objects, or copy the objects and modify the copies? The latter approach usually

is needed to produce correct simulations, but is more costly than is the former, so we analyze alternative ways of avoiding and decreasing these costs. The chapter also considers how to use frame inheritance to simplify the construction of the processes that describe chemical reactions. We present the algorithm that GENSIM uses to execute these processes efficiently, and note a constraint that must hold of process preconditions to ensure the correctness of GENSIM predictions.

1.4.3 Hypothesis Formation by Design

The hypothesis-formation methods developed in this dissertation are most similar to those labelled explanation/causal/constraint-based in Table 1.2. I treat hypothesis formation as a design problem, which is a more general and powerful approach than that used by previous researchers.

Figure 1.3 shows the relationship between the GENSIM and HYPGENE programs. Within the performance-oriented view of machine learning discussed in [Simon82,DietterichB81], GENSIM is a performance program whose task is to use its theory of molecular biology to predict the outcome of an experiment involving the trp operon. We compare GENSIM's prediction to the actual outcome of the experiment as reported in the scientific literature. If the prediction conflicts with the observation, we call on HYPGENE to improve GENSIM's performance — to improve the quality of GENSIM's predictions. HYPGENE formulates hypothetical modifications to GENSIM's theory of chemical reactions, or to what the biologists *thought* the initial conditions of the experiment were, such that GENSIM's prediction using the modified theory or modified initial conditions, matches the observation.⁶ All previous hypothesis-formation programs modify either the theory or the initial conditions, but do not provide a framework for modifying both.

HYPGENE uses the following methods to design hypotheses (Chapters 5 and 6 describe

⁶Experimental conditions are often not known with certainty by scientists.

these methods in detail). HYPGENE's initial *design goal* is to eliminate the error in GENSIM's prediction. To satisfy this goal, HYPGENE employs *design operators* that reason backward from the design goal to determine what changes to the theory or the initial conditions would achieve the goal. The operators examine a dependency trace created by GENSIM that shows how GENSIM derived its prediction from the initial conditions. The operators also examine the process knowledge base to determine what other known reactions might have occurred, and what new types of reactions are possible. The operators reason about what changes to the initial conditions would cause new reactions to occur, or would prevent predicted reactions from occurring; they also consider how to alter the processes themselves to revise the set of reactions that takes place. This reasoning is performed by a sophisticated planner that manipulates process preconditions (which can contain arbitrary predicate-calculus formulae), and process effects (which in my models are relatively procedural and thus are difficult for HYPGENE to decipher).

This design problem is a search problem because often more than one operator is relevant to satisfying a given design goal, and a single operator often can be applied in several ways. The synthetic, goal-directed search used here should prove more efficient than are the generate-and-test approaches outlined in Table 1.2. Those approaches used heuristic search to guide purely syntactic generators of hypotheses. HYPGENE uses heuristic search to guide a generator that is already focused on errors in the prediction.

I developed a novel method for guiding HYPGENE's search; the method involves a second experiment that I call a *reference experiment*. Chapter 5 shows that, if the initial conditions of the reference experiment are similar to those of the first experiment, we can sometimes attribute the prediction error to the difference between the initial conditions of the two experiments. This difference can be used both to evaluate hypotheses that HYPGENE generates, and to guide the hypothesis-generation process.

1.4.4 Results

I tested GENSIM and HYPGENE on several example problems from the historical study of attenuation (described in Chapter 7). GENSIM correctly predicted the outcomes of several biological experiments. HYPGENE formulated hypotheses to account for several anomalous experiments that the biologists performed. The outcomes of these experiments that were predicted by both the biologists and by GENSIM did not match the experimentally observed outcomes of the experiments. HYPGENE formulated many of the same alternative hypotheses to explain these experiments as did the biologists; the few cases where HYPGENE's hypotheses differed indicated interesting ways of improving HYPGENE's methods.

1.5 Readers Guide

This dissertation is long and addresses many different issues, some of which will not be of interest to every reader. Here I advise the reader how to learn about the three main topics of my work: the historical study, the representation of biological theories and their use in predicting experimental outcomes, and the formation of hypotheses.

First, Chapter 2 contains a general guide to molecular biology and a description of the *trp* operon. This information will help the reader to understand the examples discussed throughout the dissertation.

Readers who are particularly interested in the historical study should read Chapter 4. In addition, Section 5.3 and Chapter 7 describe GENSIM and HYPGENE's solutions to experiment-prediction and hypothesis-formation problems taken from the historical study. Readers who are less interested in the historical material can read the summary of the discovery of attenuation in Section 4.4.1.

Qualitative-reasoning aficionados should read Chapter 3. In addition, Section 7.1 describes

sample predictions computed by GENSIM. Readers who are less interested in qualitative reasoning than in hypothesis formation nonetheless will probably want to gain familiarity with GENSIM, since GENSIM predictions are an input to HYPGENE; they should see Section 3.5.

Chapter 5 describes my methods for hypothesis formation. Chapter 6 describes the implementation of the HYPGENE program in depth, and also provides detailed comparisons between HYPGENE and other hypothesis-formation programs. Section 7.2 describes the hypothesis-formation problems that HYPGENE has solved. Chapter 8 summarizes my hypothesis-formation techniques, and discusses their limitations.

1.6 The Evaluation of Scientific Hypothesis-Formation Programs

Since past scientific hypothesis-formation programs have been somewhat controversial, as well as difficult to evaluate, I discuss the issue of evaluating hypothesis-formation programs now so that that we keep certain points in mind in the remainder of the dissertation.

Hypothesis-formation programs are difficult to evaluate because of the nature of the tasks they attempt to perform — these tasks are “creative” and yield “discoveries.” Critics have dismissed the achievements of these programs with such statements as, “The computer wasn’t being creative; you programmed it to find that solution,” or, “The computer didn’t really discover the answer; you told it how to solve the problem.” These statements are odd because, in general, when we program a computer to do something, we attempt to “tell the computer how to solve the problem.” So it is unsettling to hear computer scientists criticize a computer’s solution to a problem on the grounds that the computer was programmed to solve that problem. What properties of creative thought might lead to such reactions?

In this age of mechanization and computerization, creative tasks are among the few types of human achievement that have not been reproduced by machines. Thus, we may equate

these achievements with our humanity, causing us to guard them jealously from all machines who would mimic them.

More important, creative processes are by their nature uncertain, and are difficult for humans to articulate and introspect about. That is, when we use a creative process to discover the solution to a problem, we have difficulty understanding how we found the solution, and we believe that whatever method we did use was not guaranteed to find a solution.⁷ Our inability to introspect about the solution process we employ may be (somewhat irrationally) responsible for some people's belief that no process actually exists.

The key property of the creative process is its uncertainty. If we believe that creative processes are by their very nature uncertain, then the idea of a creative computer program is a contradiction in terms. Computers are known to follow explicit, deterministic procedures for solving problems. But the creative process does not have these properties. So, whereas for most types of problems we consider it to be cheating to tell the computer the solution, for creative problem solving it must cheating to tell the computer *how to find* the solution, because no deterministic process can exist (we think).

Not all computer programs, however, are deterministic, or are certain to find a solution. Indeed, many hypothesis-formation programs do not contain sure-fire problem-solving techniques; they contain heuristic methods that often do not succeed. Such is the nature of search problems in general.

For many years philosophers of science sought a "logic of discovery" that would be guaranteed to produce true scientific theories. In the twentieth century, many philosophers weakened the goals of their endeavor to the search for rational methods for suggesting hypotheses, although as Buchanan points out, the term "logic of discovery" has many possible interpretations [Buchanan66].

We can draw several conclusions from this discussion:

⁷Please bear with the metaintrospection.

- Humans must be prepared to release their exclusive grip on creative thinking
- Just because we are unable to use introspection to determine the procedure we use to solve a problem does not mean that no procedure exists
- Computer programs can be as uncertain of producing discoveries as our intuition indicates people are
- We should beware of programs that promise certain solutions to uncertain problems

The preceding considerations argue that we should not "shoot first and ask questions later" on encountering a computer program that is advertised as capable of making scientific discoveries. Next we must determine what questions to ask.

The evaluation of *any* computer program should consider three characteristics of the program: the correctness and precision of the solutions it produces, the generality of its methods (the class of problems it is likely to solve), and the speed with which it produces solutions (we will ignore such properties as maintainability). These criteria are all essential for evaluating scientific-discovery programs. We can use a number of techniques to determine these characteristics for a given program. One technique is to examine sample input-output pairs for a program to determine whether that program produces correct, precise answers for a wide, representative set of problems. Another is to examine a trace of the program's execution on one or more sample problems to determine whether the intermediate results of the program are correct, general, and relatively few in number (the last criterion influences speed). Third, we can ask the programmer to describe what problem-solving method the program uses, and can use our own knowledge of the problem domain to infer the characteristics of that method.

This dissertation presents all three of these types of information about GENSIM and HYP-GENE to facilitate the reader's evaluation of these programs.

Chapter 2

Background Knowledge of Molecular Biology

This chapter presents an overview of molecular genetics to help the reader understand the many references to the domain of molecular biology in the dissertation. The discussion includes general knowledge of molecular biology, and details of a specific gene-regulation system: the tryptophan operon. It also covers experimental techniques in molecular biology. Biologists will note that certain details of biology have been omitted for the sake of brevity.

2.1 Molecular Genetics Background

The bacterium *Escherichia coli* (*E. coli*) is a single-cell organism found in the human intestine (where it facilitates digestion), and elsewhere. Biologists know more about this organism than about any other on Earth. In fact, it is likely that by the mid 1990s, biologists will know the sequence all of its DNA (its *genome*), which is roughly 1 billion base pairs in length.

The genetic material of every cell contains instructions that tell the cell how and when to synthesize the proteins and other macromolecules that constitute it. The regulation of protein synthesis is of immense importance to the cell because different proteins are needed by different

types of cells in a complex organism, and at different times during the lifecycle of a given cell. In addition, specific amounts of different proteins are needed to respond to changes in the cell's environment.

The cell synthesizes proteins from 20 different chemical building blocks called *amino acids*, of which *tryptophan* (*trp*) is one. If tryptophan and the other amino acids are present in the cell's environment, the cell can utilize them for protein synthesis. Amino acids that are not present must be synthesized by the cell from other nutrients in its environment. This synthesis is performed by enzymes (which are proteins). It is advantageous for the cell to regulate its production of different groups of enzymes that are involved in amino-acid synthesis, in response to changes in the environmental supply of amino acids. For example, the cell would be wasting resources if it synthesized the enzymes in the *trp* pathway when *trp* is present in its environment. Figure 2.1 diagrams the negative feedback loop through which *trp* concentration influences the expression of the enzymes that produce *trp*, which enzymes in turn influence the concentration of *trp*.

2.1.1 Proteins and the Genetic Material

To understand how expression of the *trp* operon is regulated, we must understand how proteins are synthesized. Protein molecules consist of long, folded chains of amino acids. More precisely, a *polypeptide* is a single chain of amino acids, and a *protein* is an aggregate of one or more polypeptides, and possibly of other molecules. The sequence of every protein in the cell is encoded by the cell's DNA, which consists of long chains of molecules called *nucleoside monophosphates*, or more simply, *bases*. There are four bases, designated A, C, G, and T. DNA is made up of two parallel chains of *complementary* bases. Bases across from each other along the parallel chains attract each other and *pair* (A pairs only with T, and G pairs only with C). For a given gene, usually only one strand encodes a protein; the other strand is meaningless.

There is some variation among the individuals of a species in the exact sequence of the

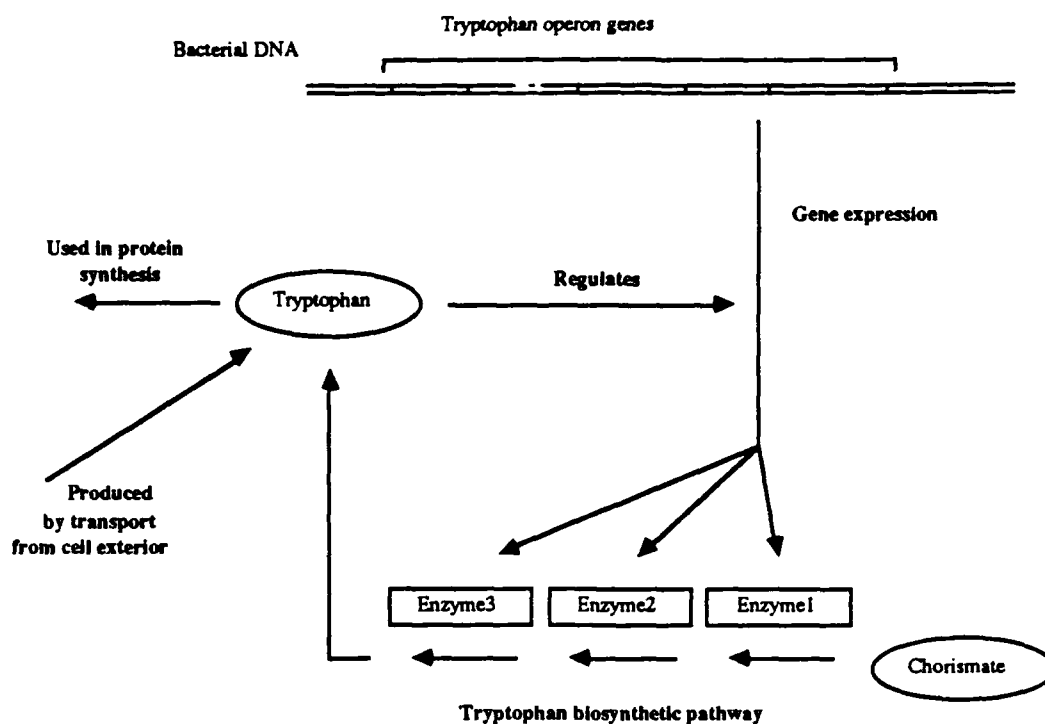


Figure 2.1: The cell regulates the intracellular concentration of tryptophan through a negative-feedback loop. The rate at which the enzymes in the *trp* biosynthetic pathway are produced is inversely proportional to the concentration of *trp*.

DNA that codes for a given protein. Most of this variation has no harmful consequences to individuals, and the individuals produce proteins that function normally. An individual that produces a normally functioning protein is said to be a *wild-type* individual with respect to that gene. Wild-type individuals thus all have the same *phenotype* (visible manifestation of a gene), although their *genotypes* (actual genetic information) may vary.

2.1.2 Transcription and Translation

The process of *protein synthesis* constructs proteins from the DNA blueprint. Figure 2.1 is a very simplified picture of protein synthesis, showing that DNA undergoes a process called *transcription* to produce an intermediate molecule called *messenger RNA* (mRNA), which is then *translated* to produce protein. mRNA is similar to DNA; however, it is single-stranded rather than double-stranded, is made up of slightly different bases (U is substituted for T), and

can be present in the cell in many copies. Because mRNA is single-stranded its bases are free to pair with other, complementary bases. Thus, it is possible for a region of mRNA to fold back on itself and to bind to an earlier complementary region of the strand. The double-stranded region thus formed is an mRNA *secondary structure*. These structures are distinctive features that can be recognized by proteins in the cell.

Transcription of DNA to mRNA is accomplished by an enzyme called *RNA polymerase*, which can recognize and bind to sequences of DNA called *promoters*. After binding, RNA polymerase moves along the DNA; it reads the DNA message and synthesizes a complementary mRNA copy of the DNA. Polymerase recognizes another site further down the DNA — called a *terminator* — that causes it to cease synthesis and to dissociate from the DNA. It is chemically possible to distinguish one end of a DNA molecule from the other. One end is called the 5' end, the other is the 3' end. RNA polymerase synthesizes mRNA in a 5'-to-3' direction. The 3' direction is also called *downstream* (and is usually to the right in diagrams of DNA); the 5' direction is *upstream* (to the left). So polymerase moves *down* the DNA. The terms *distal* and *proximal* are used as rough measures of distance within DNA — an operator-proximal segment of DNA is nearer to the operator in question than is an operator-distal segment.

Translation of mRNA to protein is accomplished by a complex system composed of the *ribosome* — an organelle (a complicated structure containing many proteins and RNA) — and various enzymes and chemical factors. The ribosome recognizes a *ribosome-binding site* on an mRNA molecule, binds to this site, and moves downstream until it encounters a *translation-start codon*, with sequence AUG. The ribosome begins synthesizing protein at the AUG codon. It then proceeds along the mRNA, reading the message in groups of three bases. Each group of three bases is termed a *codon*. The *genetic code* translates codons to amino acids and translation control signals. The RNA codon specifying trp, for example, is UGG. Figure 2.2 shows the genetic code; Figure 2.3 shows a sample sequence of DNA, and the protein it encodes.

Each time the ribosome reads a codon, it attaches the amino acid specified by the codon to

		Second Position				Third Position (3' end)
		U	C	A	G	
First Position (5' end)	U	Phe Phe Leu Leu	Ser Ser Ser Ser	Tyr Tyr Term Term	Cys Cys Term Trp	U C A G
	C	Leu Leu Leu Leu	Pro Pro Pro Pro	His His Gln Gln	Arg Arg Arg Arg	U C A G
	A	Ile Ile Ile Met	Thr Thr Thr Thr	Asn Asn Lys Lys	Ser Ser Arg Arg	U C A G
	G	Val Val Val Val	Ala Ala Ala Ala	Asp Asp Glu Glu	Gly Gly Gly Gly	U C A G

Figure 2.2: The genetic code (the mapping from RNA codons to amino acids). For example, the codon UAC specifies the amino acid tyrosine (Tyr).

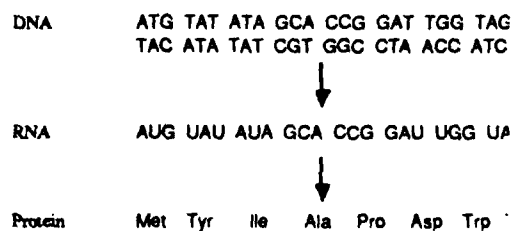


Figure 2.3: An example of protein synthesis. The process of transcription synthesizes RNA from DNA, and the process of translation synthesizes protein from RNA.

the growing protein chain. The amino acids are not floating free in the cell, but are attached to carriers called *transfer RNA* (tRNA). There is a different tRNA for every amino acid, and associated with every tRNA is an enzyme — called an *amino-acyl tRNA synthetase* — that attaches an amino acid to its tRNA. For example, tryptophanyl-tRNA-synthetase charges tRNA^{trp} with tryptophan. The ribosome recognizes another special codon called a *stop codon*, which tells it to terminate protein translation and to dissociate from the mRNA.

In an abstract sense, translation and transcription are similar processes. Both involve the synthesis of one polymer based on another template molecule (DNA into mRNA, and mRNA into protein). Both are mediated by enzymes by or enzymelike molecules. In both cases, the enzyme recognizes initiation sites (promoters, ribosome binding sites) and termination sites (terminators, stop codons) on the template molecule.

In bacteria, transcription and translation may occur on one piece of mRNA simultaneously: as an mRNA molecule is being synthesized, ribosomes may attach to it and begin protein synthesis. In fact, this concurrent translation is necessary to maintain transcription over long regions of mRNA. If transcription occurs over a long region in the absence of simultaneous translation, a mechanism called *polarity* (mediated by the *rho* protein) will terminate transcription. This safety mechanism evolved to avoid wasting resources due to mutations that might remove transcription-termination sites; polarity prevents transcription of long regions of DNA past the ends of genes that do not code for proteins.

2.1.3 Mutations

Mutant individuals are those whose DNA varies with respect to some wild-type strain. Mutants are produced naturally during evolution, and can be produced artificially in a laboratory. Functional wild-type genes are usually indicated with a "+" (for example, a functional *trpR* gene can be explicitly indicated as *trpR+*), whereas a nonfunctional mutant is referred to with a "-" (as *trpR-*). There are several different types of mutants:

- *Point* mutations are single-base substitutions in DNA; for example, a change from an A to a G
- *Insertion* mutants have one or more extra bases inserted into their DNA
- *Deletion* mutants have one or more bases removed from their DNA

Consider how the three-base codons can be affected by insertion and deletion mutants. If the number of bases inserted or deleted is a multiple of three, then the codon *reading frame* is not altered, so only a small, probably insignificant change has been made to the protein. But if the size of the insertion or deletion is not a multiple of three, then the entire reading frame after the mutation is altered, changing the remainder of the protein. Such an insertion or deletion is called a *frameshift* mutation. Mutations can also cause base substitutions; a substitution that alters the amino acid coded for by a codon is termed a *missense* mutation. A substitution that changes a codon for an amino acid into a translation stop signal, or viceversa, is called a *nonsense* mutation. Sometimes a second mutation may cancel out the effects of an initial mutation and produce a wild-type organism. Such an organism is termed a *revertant*.

Virtually all of *E. coli*'s genes reside on one chromosome (one long piece of DNA), although small circular DNA molecules called *plasmids* can also carry several genes, and can be injected into and transferred among different bacterial cells.

2.1.4 Regulation

An *operon* is a set of genes whose synthesis is regulated as one unit. Usually, the proteins coded for by an operon have related functions. The *trp* operon contains five genes, which code for five polypeptides. One of the five encodes one enzyme; two pairs of two polypeptides bind together to form the other two *trp* enzymes. There is a promoter at the start of the *trp* operon, and a terminator at the end. The resulting mRNA codes for all five polypeptides, and thus contains five ribosome-binding sites, five translation-start codons, and five stop codons.

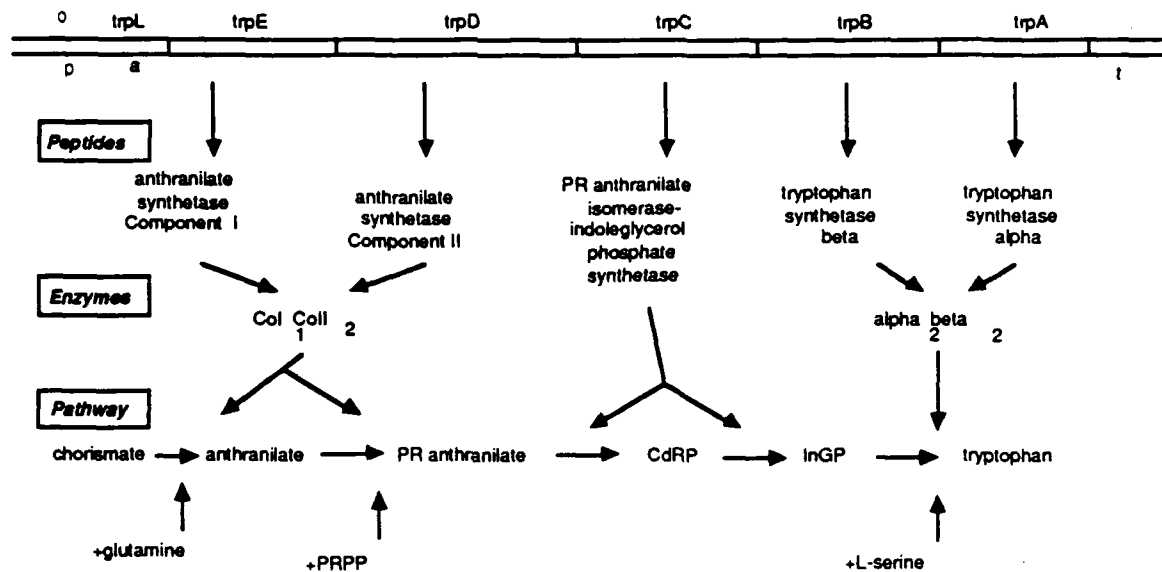


Figure 2.4: The *trp* operon contains five structural genes: *trpA* through *trpE*. The genes are surrounded by genetic control regions called the *promoter* (p), *operator* (o), and *terminator* (t). The operon also contains a region of DNA called the *leader region* (*trpL*). The genes in the *trp* operon code for five polypeptides that form three enzymes. These enzymes form a biosynthetic pathway that synthesizes trp from chorismate and other chemical precursors.

Figure 2.4 shows the chemical reactions in the *biosynthetic pathway* for trp, the enzymes that catalyze these reactions, and the layout within the *trp* operon of the genes that code for these enzymes.

The cell regulates the process of transcription through *Jacob-Monod repression*. This regulation mechanism was discovered in the late 1950s in the *lac* operon [JacobM61]. Within the *trp* promoter is a region called the *trp operator*. A protein called the *trp-repressor* can recognize and bind to the *trp operator*. When bound there RNA polymerase is unable to bind to the *trp* promoter. The *trp-repressor* protein contains another site, to which trp can bind. When trp is *not* bound to *trp-repressor*, the protein is unable to recognize the *trp operator*, and thus cannot interfere with transcription. But when trp is bound to the repressor, the protein can bind to the operator. In this way, transcription of the *trp* operon varies inversely with the concentration of trp in the cell: the cell produces more of the *trp* enzymes when less

trp is present in the cell's environment.

When *E. coli* is grown in the presence of trp, the trp-repressor protein is *activated* and the operon is in a *repressed* state. When no trp is present, the operon is *derepressed*. Experimenters can construct mutant cells that have nonfunctional trp repressor proteins, termed *trpR*-. These cells cannot regulate the synthesis of the trp enzymes and are said to express the operon *constitutively*.

The preceding description summarizes what biologists knew about the trp operon in the early 1960s. To enable the reader to understand more easily the descriptions of the scientific research that follow, I shall now describe what was wrong with the above theory.

Although the trp operon is indeed regulated by repression, it is also regulated by a second, independent mechanism called *attenuation*. The region of DNA in between the trp operator and the start of the *trpE* gene (see Figure 2.4), called the *leader region*, contains a DNA site called the *attenuator*. As RNA polymerase synthesizes the leader-region mRNA during transcription, the attenuator sometimes signals RNA polymerase to prematurely terminate transcription. The frequency of termination depends on the concentration of trp within the cell. At high trp concentrations, the attenuator causes frequent termination of transcription (an attenuation of the transcription process), whereas when trp concentration is low, RNA polymerase usually *reads through* the attenuator and continues transcription. A low trp concentration *relieves* attenuation. Because the trp operon is influenced by two regulation mechanisms, its output can be regulated over a range that is wider than a single mechanism would allow.

In summary, the cell produces proteins from information in DNA via two linked processes: transcription and translation. Two mechanisms regulate protein synthesis at the transcriptional level: repression controls the rate of transcription initiation, and attenuation controls the rate of premature termination of transcription. Both mechanisms are sensitive to the concentration of trp, so when there is less trp present in the cell, the cell manufactures the trp biosynthetic enzymes to synthesize tryptophan.

2.2 Techniques for Research on the trp Operon

The experimental techniques available to molecular biologists in 1968 were important in determining what experiments were performed on the trp operon. They therefore had a strong influence on what theories of the trp operon would be generated and tested. The knowledge that the group possessed about the trp system at that time was closely tied to the techniques that were available to the group, since this knowledge was gleaned from prior application of these (and older) techniques. This section describes these experimental techniques, and those that became available over the next 10 years or so. Powerful new techniques substantially altered the experimental questions that could be answered, and the speed with which answers were obtained.

Two important experimental concepts are those of *in vivo* experiments and *in vitro* experiments. *In vivo* experiments involve measurements on an actual organism, such as measuring the enzyme production of a given strain of *E. coli*. *In vitro* experiments strive to reproduce some part of the cell's machinery within a test tube, to study it in isolation. If we can duplicate some part of the cell's machinery or some cellular event with laboratory reagents, we have probably identified all the components of that machinery.

The experimental techniques are grouped into two classes: observational and manipulative techniques. The former allow a biologist to measure some aspect of the trp system. The latter are used to fabricate an organism or an *in vitro* experiment with desired qualities. Manipulation is important because standard paradigms of biology are to perturb, dissect, reconstruct, and disable parts of a system under study in order to determine how the parts contribute to the function of the system as a whole. These two classes of techniques are usually used together in the laboratory: a manipulation is useful only if its effects can be observed, and the power of observational techniques can often be amplified if certain manipulations are performed.

Observational technique	Used to Identify
Observation of cell growth or death	Presence of toxins or absence of required nutrients
Biochemical assays	Quantities of chemicals present
DNA sequencing	DNA sequence
RNA hybridization	Presence of specific RNA species
RNA fingerprinting	Presence of specific RNA species
Enzymatic digestion	Binding sites between proteins and RNA or DNA, and RNA or DNA secondary structures
Pulse labeling, sampling	Time course of a biochemical process
Observation in electron microscope	RNA or DNA secondary structures
Recombination	Chromosomal distances between genes
Restriction enzyme mapping	Presence of specific DNA species

Table 2.1: Different observational techniques, and the entities they measure

2.2.1 Observational Techniques

This section outlines the observational techniques and the different quantities within the *trp* system that they measure (see Table 2.1).

A basic observation is to determine if a bacterial strain is able to grow in a given medium, as evidenced by a visually observable bacterial colony, or lack thereof. Cells will reproduce if they are able to utilize any nutrients in the medium (such as *trp*), and if they are resistant to any poisons that might be present.

The presence of many chemicals in the cell can be measured through a variety of assays. For example, we can measure the presence of tryptophan, of total protein, of specific proteins

such as the trp enzymes, and of mRNA. Often considerable effort was expended to develop new assay procedures for entities whose presence could not be quantified.

Before 1975, it was practical to obtain the sequence of only short lengths (roughly 100 bases) of mRNA. The DNA-sequencing technology developed in 1975 revolutionized this field by allowing researchers to sequence segments of DNA that were thousands of bases long in days of work. With sequence in hand biologists could determine the precise locations of genes and control regions, discover previously unknown functional elements using computer-based pattern matching, and map mutations precisely.

Hybridization techniques allow us to isolate a specific *species* of mRNA (such as a particular gene) to quantify its presence in a culture. More precisely, the desired species will base pair and adhere to previously collected samples of the DNA species of interest, which have been bound to some substrate material. mRNA and other macromolecules can also be filtered in gels to separate different species from one another. *mRNA fingerprinting* is another way of identifying the presence of specific mRNA species within a culture: we use enzymes to digest the mRNA into a set of large fragments, and the size distribution of these fragments forms a "fingerprint" that can be used to identify the same mRNA species — or a substrand of it — in other experiments.

Digestion reagents can be applied to DNA or RNA that is bound to a protein to locate the site to which the protein is bound. All unbound DNA/RNA is digested away, leaving the binding site intact.

Pulse-labelling and *sampling* techniques allow us to quantify the presence of a molecular species in a culture over time, or within precise intervals of time. Measuring the time course of a process often helps to distinguish among different possible mechanisms for that process.

The visual structure of macromolecules can be determined in electron microscopes, which allows experimenters to detect secondary structures (by visually observing a folded molecule) and binding sites (by observing contact between two molecules).

Manipulative technique	Used to Create
Mutagenesis, screening	Bacteria with desired traits
Supply or deprive medium of nutrients, analogue*	Desired environment for bacterial growth
In vitro experiments	Subsets of cellular machinery
Recombination	Bacteria with desired genetic make-up
Restriction-enzyme techniques	Bacteria with desired genetic make-up

Table 2.2: Different manipulative techniques, and the conditions they are used to tailor

Recombination experiments combine the DNA from two parental bacteria and generate daughter strains whose DNA is some combination of the parental DNA. The frequency at which a recombinant strain is produced allows us to infer the relative location of mutations within the chromosome to a resolution of 1 base (used before DNA sequencing techniques became practical).

Restriction enzymes were discovered in the mid-1970s. They selectively cut DNA at specific base patterns, and are useful in generating maps of long DNA regions, forming a kind of DNA fingerprint.

2.2.2 Manipulative Techniques

A researcher uses manipulative techniques to tailor specific experimental conditions. The important manipulative techniques available to molecular biologists are summarized in Table 2.2.

To produce bacteria with desired traits (such as a nonfunctional *trp* promoter) we generate a large number of random mutant bacteria using radiation, then screen for those cells with the desired characteristics (by killing cells that lack the desired traits). The precise location of the mutations in a selected strain can be determined using recombination experiments or

sequencing, to verify that the phenotype is in fact caused by a nonfunctional trp promoter.

The environment of a cell culture can be manipulated in many ways. A culture can be supplied with or deprived of nutrients such as tryptophan or sugars. Sometimes, chemical *analogs* of nutrients can be found that have behaviors slightly different from those of the nutrients themselves. For example, an analog of trp called 5-methyl-tryptophan will bind to the trp repressor to repress the trp operon, but cannot be used in protein synthesis; normal cells will not grow in the presence of 5-methyl-tryptophan. The temperature of the environment can be manipulated to alter reaction rates and to screen for temperature-sensitive mutants.

In vitro transcription and translation systems can be assembled and perturbed. It can be technically difficult to construct a given in vitro system, because some cell processes are complex or are incompletely understood.

Cis-trans experiments are used to decide whether regulatory control is mediated by diffusible entities such as proteins, or by effects local to one region of DNA, such as attenuation. In these experiments, a plasmid containing a copy of a gene or other DNA segment of interest is introduced into a cell. Often the cell contains a mutant version of the same DNA segment. Depending on whether or not the behavior of the cell is changed by the new plasmid, we can infer whether or not the added DNA codes for a diffusible factor (such as a repressor protein), or if it must be adjacent to the DNA it controls (such as an operator). A *dominant* effect occurs when the introduction of a plasmid containing a wild-type gene restores wild-type function to a mutant cell.

The following techniques were earlier categorized as observational, but they can be considered manipulative as well. *Recombination* experiments can construct a bacterial strain with a desired genotype by combining the DNA from two parental bacteria and generating daughter strains whose DNA is some combination of the parental DNA. *Restriction enzymes* can be used to extract specific regions of DNA from one or more cells, which can then be spliced together to produce "designer genes."

2.2.3 An Example Experiment

We now illustrate the use of some of the preceding techniques through an example.

Imagine that we wish to measure the production of *trp*-mRNA in constitutive operator-mutants of the *E. coli* *trp* operon (which we shall designate a *trpO*- strain).

The first step is to obtain the *trpO*- strain itself. To do so, we expose a population of cells to radiation to generate random mutants, then grow these mutants in 5-methyl-tryptophan. As mentioned previously, 5-methyl-tryptophan kills normal cells because it represses their *trp* operons, and it cannot be used for protein synthesis. These cells have no *trp* on which to grow.

Among the cells that can grow are mutants whose *trp* operons are always expressed — constitutive mutants. This phenotype could be caused by several specific types of mutations, including mutations in *trpR* and *trpO*. To distinguish the desired *trpO* mutants from all others, we map the locations of the mutations using recombination techniques.

These manipulations yield a mutant strain whose mutation maps to the vicinity of *trpO*. Now the strain can be characterized. It should be grown in both *trp*-free and *trp*-excess media, then killed at specific intervals. The mRNA synthesized by the strain can be isolated, and the *trp*-mRNA can be selected by hybridization to previously collected *trp*-mRNA. We can measure the amount of *trp*-mRNA present by growing these cells in radioactive mRNA precursors, and counting the radioactivity within the hybridized *trp*-mRNA.

2.2.4 The Manipulation Versus Observation Distinction, Reconsidered

Although the distinction between observation and manipulation can be valuable for thinking about these techniques, it is also instructive to blur this distinction. As manipulative techniques are applied, observations must be made periodically to verify that the manipulations have succeeded, and to select among populations of cells that have responded differently to a given manipulation. Similarly, the process of observation often necessitates different types of manipulations to render an entity observable. Even the use of a microscope could be considered

a manipulative technique: Objects must be specially prepared for viewing: they are washed in several chemical solutions, sliced on a special machine for viewing, and mounted on a glass slide. Once they are under the microscope, we manipulate the relative positions of the lenses and slide to enlarge and focus the image (but to blur the distinction), and to position different parts of the image for viewing.

2.3 Summary

This chapter provided a general overview of molecular biology, and a description of the tryptophan-operon gene-regulation system. The chapter also presented many of experimental techniques that biologists used to study the regulation of the trp operon. The reader will find this information helpful in understanding many of the examples in this dissertation.

Chapter 3

Declarative Device Models of the Tryptophan Operon

The task of using computers to formulate scientific theories has an important prerequisite. We must be able to represent scientific theories within the computer in such a way that they can be used to predict outcomes of scientific experiments, and such that they can be reasoned about and modified by a hypothesis formation program. This chapter presents techniques for representing theories of molecular biology, for representing experiments in molecular biology, and for using theories to predict the outcomes of experiments. Three different models of the trp operon were implemented in the course of this thesis research. The models have different capabilities that reflect that each was constructed as an experiment with different AI techniques.

Section 3.1 describes general characteristics of scientific theories. Section 3.2 rejects existing terminology in the field of "qualitative reasoning," and proposes that the term *declarative device modeling* is more appropriate. Section 3.3 describes the first model of the trp operon, which uses KEE frames to describe biological objects, and KEE rules to describe chemical reactions between these objects. The first model has severe limitations.

The second model is described in Section 3.4. It does not predict what reactions occur in a given experiment, but is concerned with predicting reaction rates in a reaction network that it is given. This model incorporates new techniques for qualitative reasoning about mathematical functions that are useful when only partial knowledge is available about the values of system state-variables, or about the mathematical functions that describe dependencies among these state variables.

Section 3.5 describes the third model, called GENSIM. It embodies a *qualitative chemistry* — an ontology for chemical objects and chemical reactions. This qualitative chemistry provides a framework for simulating chemical reactions. GENSIM is the model used in conjunction with the HYPGENE hypothesis-formation program; HYPGENE does not interact with the earlier models. (Readers who are less interested in general issues in declarative device modeling, but who wish to learn more about GENSIM to increase their understanding of HYPGENE, should skip to Section 3.5.) The key features of GENSIM are

- Its chemical objects correspond to *populations* of molecules. An assumption-based truth-maintenance system (ATMS) can be used to make the representation of the objects generated during a simulation more compact and efficient.
- Chemical reactions between these populations are probabilistic phenomena. To simulate them correctly, we must view reactions as events that *fork* populations into two parts: a subpopulation that participates in the reaction, and a subpopulation that does not react.
- We can increase simulation efficiency by merging identical objects that are synthesized during a simulation.
- The framework specifies restrictions on the syntax of preconditions for chemical reactions that are necessary to ensure the correctness of simulations. Although the restrictions were derived from an analysis of the GENSIM program, they have a valid chemical interpretation.

The framework that GENSIM defines for representing scientific theories is able to represent a wide range of scientific theories in molecular biology, and is likely to be useful in other domains as well. This framework utilizes Forbus-style processes rather than the fixed state-variable network employed by deKleer and Brown; we argue that the former is considerably more flexible.

3.1 Scientific Theories

A scientific theory is a device for prediction. A theory is a computational entity that takes as input a description of a system (such as a bacterial cell) at time t , and produces as output a description of the system at a later time t' . The input describes the initial conditions of a scientific experiment, and the output predicts the outcome of the experiment. Ideally, the predictions of the theory will always match the observed outcome of an experiment.

Some experiments have a more general form: the experimenters both establish initial conditions and perturb the system at later times, such as by adding reagents to an ongoing experiment. The preceding framework would treat such an experiment as a series of computations, with a new computation beginning each time the experimenters alter the physical system in some way.

Within this perspective, the theory of the trp operon is viewed as a device for predicting the outcomes of experiments in the trp operon.

Some theories make another type of prediction as well — namely predictions as to the structural arrangements of objects that are likely to be found in physical systems. For example, the repression theory of gene regulation asserts that operons are likely to be found in bacterial DNA, and that an operon is a set of one or more genes (each including a ribosome-binding site, start codon, and stop codon) surrounded by a promoter-operator region and a transcription terminator. The theory describes classes of biological objects and specifies the properties of

these objects. For example, objects contain other objects as parts, and have attributes such as charge and concentration.

Theories make predictions by describing how objects interact over time. These interactions are dependent on what objects are present in an experiment, on the properties that these objects have, and on general properties of the environment such as temperature. Interactions in the trp system are chemical events, such as the binding of two molecules or the series of reactions involved in the transcription of DNA.

Theories are associated with one or more *domains of applicability* — those portions of the physical world in which the predictions of the theory are expected to be valid. For example, theories of the trp system are applicable to predicting the rate of expression of the *E. coli* trp operon in certain growth media, but not to predicting the reproductive success of the horned toad in downtown San Jose.

Finally, theories are usually assigned a degree of credibility — a measure of one's confidence in the predictions of the theory. The credibility of a theory may be dependent on the theory's domain of applicability, since in general our confidence in a theory's predictions decreases as the theory is employed further and further from its intended domain of applicability.

3.2 Terminology

So many terms have been used of late to describe the research problems with which this chapter is concerned that it is worthwhile to attempt to establish some meaningful terminology before we are suffocated by inappropriate names for this field.

I propose the term *declarative device modeling*. This phrase captures the viewpoint that this subfield of AI takes of a variety of systems. Be they electrical, mechanical, biological, economic, chemical, sociological, or ecological, we wish to view them as *devices* — to describe their structure and function so that we may predict their future behavior, explain their past

behavior, diagnose their current behavior, and design new devices that replace or incorporate them. Having a device *model* is key because the description of the structure and function of the device is what is common to all these tasks. The model must be in a *declarative* form so that the entities that perform the preceding tasks are able to reason about the model.

Among the existing terms for this subfield are *qualitative reasoning*, *qualitative physics*, *qualitative simulation*, *causal reasoning*, *causal modeling*, and *commonsense reasoning*. Each has specific drawbacks. *Simulation* is troublesome because technically it means prediction, which is a subset of the tasks in which we are interested (some researchers use this word inaccurately to describe all the preceding tasks). The word *causal* is enticing, but its use is often presumptuous because we have yet to see declarative device models that include a rich, declarative, comprehensive, meaningful description of causal as opposed to other types of relationships, and that are based on a precise theory of causality. The term *qualitative physics* is overly specific because even though some researchers are immediately concerned with the domain of physics, their techniques are usually applicable to other domains, and modeling techniques developed for other domains are often relevant to qualitative physics.

The term *qualitative* is not as general as some authors' use of the word implies: We can build declarative device models that are not qualitative, and qualitative models that are not declarative. The word *qualitative* usually indicates that a model incorporates a technique for abstracting mathematical relationships or state variables, such as the use of confluences, limit analysis, or aggregation. A declarative device model may or may not include such abstractions. Similarly, a model that does include such abstractions may or may not be in a declarative form. For example, Davis and Genesereth [Davis84, Genesereth84] have constructed declarative device models that are not qualitative; they do not use special representations that abstract either the state variables or the component interactions in the devices they model.

3.3 Model 1: The Rule Model

The first and simplest model I constructed employed KEE frames to represent objects and KEE production rules to represent interactions between objects. This model was a prototype that focused on the processes involved in transcription initiation.

KEE units represent objects such as the trp-repressor protein and tryptophan. As well as being an instance of the class of proteins, trp-repressor is also an instance of a class of objects called *molecular switches*. A molecular switch is a member of a general class of molecules that change their state when another molecule — called the *cofactor* — attaches to them. Tryptophan is the cofactor for the trp-repressor. The behavior of molecular switches is captured by a single production rule that can be applied to the several types of molecular switches that exist in the model (the trp operator is a molecular switch; its cofactor is trp-repressor).

Model 1 is interesting in that it provides an example of a program that uses production rules to represent the structure and function of a physical system. As discussed in [KarpW88], some authors have argued that such “shallow” expert-systems techniques cannot be used to construct causal models.

Similar models of gene-regulation systems are described in [Meyers84,Koton85,Weld86,Round87]. Meyers constructed a system that models the life cycle of *Lambda Phage* (a virus). It models the structure of the phage using frames; the behaviors of objects in the Lambda system are represented using production rules. Meyers’s program iteratively determines what reactions occur in the Lambda system (such as expression of its genes and degradation of its repressor protein) at successive points in time. Reaction rates and protein concentrations are represented quantitatively in a manner that is intuitively reasonable, but is not solidly grounded in experimental data — the accuracy of the numbers in the simulation does not justify their precision. The program can predict the behavior of Lambda when mutations are introduced

into different regions of the Lambda DNA. This model describes a very small number of objects and reactions. Round's model of the *E. coli* trp operon is similar. Objects in the trp operon are described using KEE frames, but Round developed a process-description language for describing the behaviors of these objects. It allows one to decompose processes into smaller subprocesses. It has a quantitative component that is essentially the same as Meyers'. Round's model also provides graphical animation capabilities.

Weld has addressed the problem of using a model of a single chemical reaction to predict the aggregate behavior of many molecules that undergo the same reaction. His technique is similar to mathematical induction. For example, the *aggregation* technique allows him to predict the transcription of an entire gene by reasoning about an individual transcription-elongation event that advances RNA polymerase one base along a DNA strand. This technique can solve certain problems in this domain, but it has an important limitation: Weld's program cannot predict the final sequence of the transcribed RNA because the aggregation technique does not copy every base from the DNA to the RNA; it merely states that an RNA of the same length as the DNA will be produced. For similar reasons, his program would be unable to predict what proteins would be translated from a given mRNA (Koton also made this observation [Koton85]). Koton's model is described in Chapter 6.

My model 1 proved adequate for simulating the cascade of reactions that regulate transcription initiation, but it and the preceding models of other researchers have six limitations. First, some models have no notion of quantity and thus cannot predict at what rate these reactions occur, nor what amounts of reactant products accumulate. Second, some models use purely quantitative notions of quantity when in fact only qualitative knowledge about quantities is usually available in this domain (and usually only qualitative predictions are desired). Third, these models are generally not able to represent the complex part-whole hierarchies that exist in the trp system. For example, the trp operon has a number of parts, such as the promoter and operator. We wish to not only represent the component structures of particular objects,

but to represent the structures of classes of objects, and to automatically instantiate these classes in a particular experiment (see Section 3.5.2). Fourth, the production rules used to represent object behaviors lack expressiveness: GENSIM processes allow negation, disjunction, and quantification in their preconditions, and these constructs are needed to define certain reactions properly. Fifth, Section 3.5.6 shows that in order to simulate chemical processes correctly, it is necessary to copy reacting objects and modify the copies, rather than to modify the objects directly. This issue of *object forking* ensures simulation correctness, but introduces a number of complications. The models discussed in this section do not address this issue. Sixth, we wish to increase the complexity of the models by encoding descriptions of more objects and more reactions, and by increasing the detail of these descriptions. Models 2 and 3 address all of these issues, plus several others.

3.4 Model 2: The Fixed State-Variable Model

Model 1 focused on describing objects using frames, and on describing how object states change due to interactions among objects. The second model ignores most object states and focuses on quantitative aspects of the trp operon.¹ Whereas model 1 is concerned with predicting *what* changes occur in an experiment, the second model is concerned with the *rates* at which changes occur and with the *amounts* of objects that are produced and consumed by chemical reactions. The second model contains only knowledge of quantitative state variables of the trp system, such as the concentration of tryptophan. It knows neither that the trp-repressor is a protein, nor what the part-whole structure of the repressor is.

I approached these problems from the perspective of qualitative simulation (which I term *declarative device modeling*, or *DDM*) — as explored in [deKleer84, Forbus84, Davis84, Kuipers84]. Most of these researchers are concerned with the properties of mechanical systems such as pipes, refrigerators, and bouncing balls. We are concerned with biochemical systems that

¹This model was developed jointly with Peter Friedland.

contain populations of interacting molecules. These physical systems are similar in that they can all be described mathematically, and qualitative reasoning techniques apply to most of these mathematical descriptions. Because our simulations take place in a complex, real-world domain in which we desire close to expert-level predictions, we have identified limitations of the existing techniques, in addition to those that have previously been identified [Kuipers85].

In the domain of gene regulation we found that the form of the knowledge available for problem solving varies widely in its precision from quantitative to qualitative. We considered constructing a model using the techniques developed by deKleer and Brown [deKleer84], but realized that the predictions it would be capable of making would be much coarser than are those an expert biologist is able to make. The predictions would lack precision because such a system would lack much of the knowledge that a biologist uses. Our goal has been to attempt to bring more diverse types of knowledge to bear on this simulation problem than the representations developed by other researchers allow. This goal has led us to develop new techniques and representations for simulation that span a continuum from quantitative to qualitative.

3.4.1 Techniques

In constructing this model of the biological domain, our first goal was to understand as much as possible about the problem-solving methodologies of molecular biologists. This process included extensive interviews with Professor Yanofsky and our other collaborators, and reviews of much of the relevant (and voluminous) scientific literature.

An important intermediate point in this process led us to develop the techniques used to construct model 2. At this point, we the computer scientists possessed a very qualitative understanding of the biological system: We knew what objects existed in the system, what properties these objects had to have to interact in certain ways, and what new objects were created as a result of these interactions. It became clear, however, that the biologists could

make more specific predictions about the system than we could. For example, biologists can predict the amount of enzyme produced when the concentration of tryptophan is 0, or when it is at an equilibrium value, or at some small constant times the equilibrium value (for example, one-half of or twice the equilibrium value). Thus, the biologists possessed quantitative information of the trp system that we had not yet acquired.

We knew that the biologists did not have a complete quantitative understanding of the system: Some state variables of the system had never been measured, or were known very imprecisely or under a narrow range of conditions. Also, the functional dependencies between many state variables were not known precisely. In summary, the biologists' knowledge lay between the lower bound of our completely logical understanding of the system, and the upper bound of a complete quantitative description of the system. We sought to understand just what knowledge the biologists had, and how they employed it to predict the results of experiments.

We shall describe representations and reasoning techniques we developed to encode and use this knowledge. The principal components of this model are not objects, but rather are state variables of the trp system that represent quantitative aspects of the system, such as concentrations and reaction rates. Figure 3.1 shows the network of state variables and their interactions that we have constructed for the trp system. Our examples will focus on two molecular binding processes within this network. In the first, molecules of tryptophan bind to molecules of the trp repressor-protein to *activate* this protein. We thus refer to the concentration of tryptophan in the cell (the variable `trp`), the total concentration of repressor protein present (`Total.trpR`), and the concentration of activated repressor protein (`Activated.trpR`). Similarly, the molecule RNA Polymerase can repeatedly bind to a DNA site called a promoter when it is free — an event called *transcription initiation*. We thus refer to the amount of `RNA-Polymerase`, the `Free-Promoter.Lifetime`, and the `Transcription.Initiation.Rate`.

Like most other approaches to DDM, ours includes three components: a means of representing state variables of the system, a means of representing relationships between these state

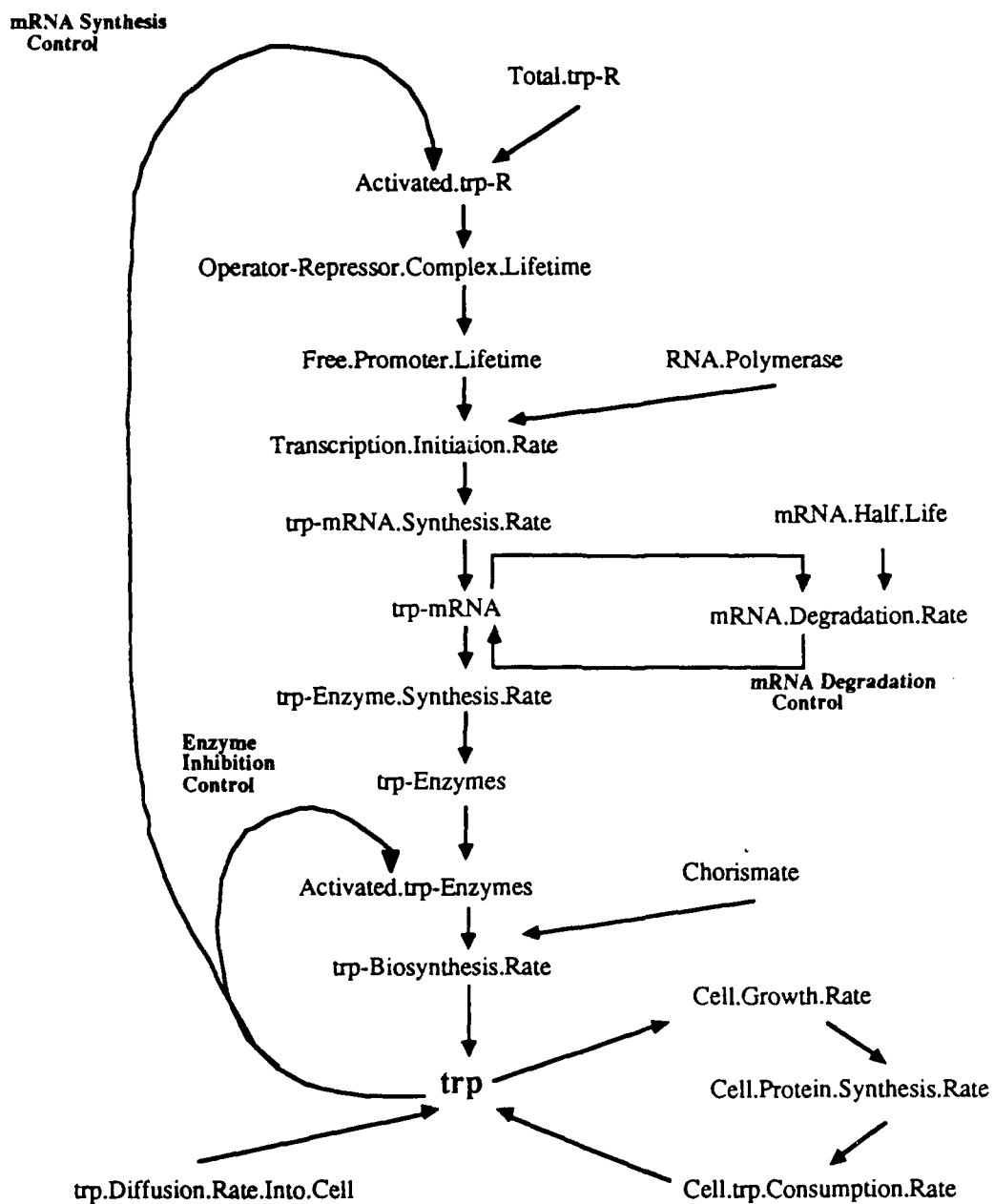


Figure 3.1: A fixed state-variable network that describes the trp operon system.

variables, and a means of reasoning about these state variables and relationships to produce a prediction and explanation of behavior. We shall describe our representations and simulation procedure, and shall contrast our design with previous approaches.

We encode state-variable values and dependencies among state variables using a combination of existing representations and representations that we have developed. It became apparent that multiple representations were necessary when we observed that biologists are often unable to determine both the exact quantitative values of state variables, and the precise mathematical relationships among the variables. They are, however, able to make several different types of useful statements about these values and relationships.

State-Variable Representation

Our representation is designed to capture a variety of types of assertions about the value of a variable. The value of a variable at a point in time during a simulation is called a *value instance*, or *vin* for short. Each vin has a name based on the name of the associated variable. In addition, several types of assertions can be made about the value of a vin, including

- Assertions with respect to quantitative values,² such as $[trp = .001]$,
 $[trp > .005]$
- Assertions with respect to other vins, such as $[trp = trp.maximum]$, $[trp > trp.equilibrium]$
- Relative assertions, such as $[trp = 2 * trp.equilibrium]$,
 $[trp = trp.maximum - .0001]$

Any combination of such assertions may be made about a given vin. The syntax of allowed assertions is: $[vin \text{ relop } vin]$ and $[vin \text{ relop } vin \text{ op } vin]$, where *relop* is a relational operator such as "<" and *op* is an operator from the set $\{+, -, *, /\}$.

²The system does not reason about units of measurement; all units are assumed to be molar.

These representations are similar to those in Simmons' quantity lattice, used to model geologic processes [Simmons86]. The quantity lattice records assertions about many interrelated values. They are stored in a graph whose nodes are algebraic expressions and whose edges are relations. In response to queries, the quantity lattice determines whether some relationship holds between two values by searching for paths between the two graph nodes that represent the values in the query, and analyzing the paths to determine whether the queried relation holds.

We have adopted a variant of his implementation and inference techniques. We use bidirectional search to find a path between two quantities in the lattice, which is an efficient search technique for this problem. We are using a subset of Simmons' techniques in that we cannot represent relationships among arbitrary expressions such as $[A + B > C + D]$. This restriction simplifies the implementation, and excludes the more computationally complex technique of constant elimination arithmetic. Thus far, we have not required the more complex types of inferences in our system. We have noted that it may often be necessary to describe a quantity using several intervals rather than only one, because only disjunctive information may be available about a quantity. For example, the statement $[A \neq B]$ is really a disjunctive statement that $[(A < B) \vee (A > B)]$. Additional disjunctive information could give rise to additional intervals. We have not implemented this type of disjunctive reasoning.

The fact that we independently arrived at such similar value representations from completely different domains is empirical evidence of the utility and generality of these representations.

DeKleer and Brown have experimented with a qualitative physics in which a variable may have a value from the set $\{-, 0, +\}$ [deKleer84]. As mentioned, this representation is unable to express a host of important distinctions that we find to be necessary for producing expert-level predictions. Kuipers uses a less coarse representation called *landmark values*, which provide a way of naming values of interest that different variables may take, and of maintaining a partial

ordering among these values [Kuipers84]. Our value representations are a superset of both Kuipers' and deKleer and Brown's.

Langlotz discusses the use of yet another type of value representation, namely probability distributions [Langlotz86]. This technique encompasses all those we have discussed, since different probability distributions could be used to represent any of the preceding types of value information. For our domain, however, we see two drawbacks to the use of probability distributions: their computational complexity may be prohibitive, it is not clear that it is possible to elicit such detailed information from experts, and we often do not required as detailed predictions as probability distributions produce. Generalizing from these comparisons, we assert that these three dimensions (computational complexity, ease of elicitation from experts, and required precision of the solution) are the crucial ones for describing a qualitative representation, and for evaluating its applicability to a given task domain ³.

Representation of Interactions

Just as there is a range in the degree of precision with which we might know the value of a given variable, there is an analogous range in the precision with which we might know the mathematical relationship between two variables. Although *some* function describes how interacting state variables in the system influence one another, biologists may not have been able to determine the exact behavior of each function. Even if experimenters have detailed information about the behavior of a function, they may be unable to determine its algebraic form. Yet, experiments and theory may provide *some* information about the properties of the function.

Thus, we require representations that allow us to represent the information we have about a function, even if it is only approximate information. These representations of the function can differ in two orthogonal ways from the actual function: in precision and in accuracy. A *less*

³Curtis Langlotz contributed to my understanding of the importance of these dimensions.

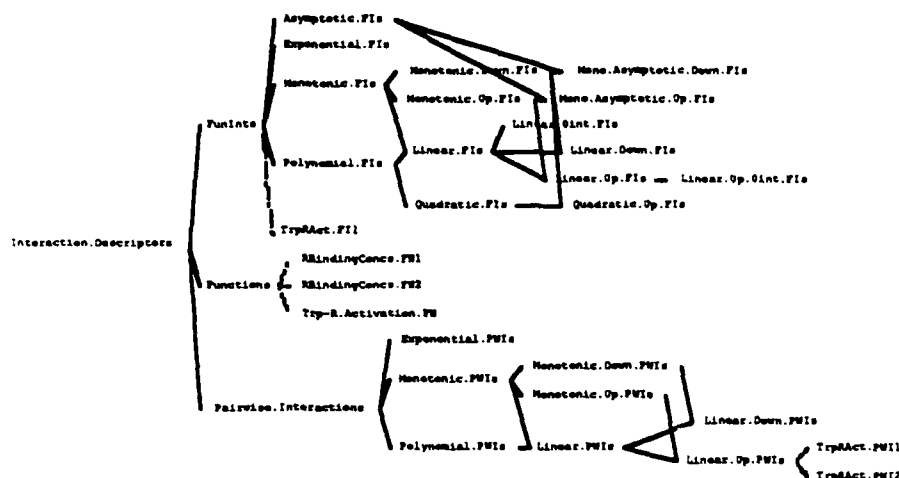


Figure 3.2: A generalization hierarchy of mathematical functions. Functions on the right of the figure have more constrained behaviors.

precise representation blurs the mapping from the function's inputs to its outputs, resulting in a more qualitative output, but one that is not incorrect. A *less accurate* representation can produce a precise, exact mapping from inputs to outputs, but one that may be incorrect to some degree. Figure 3.2 shows a hierarchy of function classes that demonstrates this idea.

This section describes how interactions among variables are represented; Section 3.4.1 describes how and when these representations are used.

We consider functional relationships among variables to be complex concepts; they are represented with several frames, within which all or only some slots may actually be filled (since only partial knowledge about an interaction may exist). First, functional relationships between each *pair* of interacting variables are represented with frames called *pairwise interactions*. The slots within these frames allow us to record all or any subset of the following information about a functional relationship between two variables x and y :

- The monotonicity of the pairwise interaction, that is, if the sign of $\partial y / \partial x$ constant
- The sign of $\partial y / \partial x$ (if it is constant)
- The form of the functional relationship between x and y — linear, polynomial, exponential, unknown

- The sign or magnitude of the exponent of a polynomial or exponential function
- The sign or magnitude of the coefficient for the interaction

For example, a pairwise interaction would describe the relationship between `Free-Promoter.Life` and `Transcription.Initiation.Rate`.

Frames called *functions* describe how a *set* of input variables combine to influence an output variable. All the combinations in our system are additive, multiplicative, or unknown. A function frame can reference one or more *mapping* frames (mappings describe observed values of the function). A function would be used to describe how `RNA-Polymerase` and `Free-Promoter.Lifetime` combine to influence `Transcription.Initiation.Rate`.

Notice that if all the slots in the pairwise interaction and function units describing a given relationship are filled, we have the ability to describe a quantitative algebraic constraint. If the values of numeric constants are omitted, we are left with qualitative constraints. If even less information is specified, we can represent even less precise interactions (such as recording that a function is linear and monotonically increasing).

The motivation behind our use of mappings is that even if we do not know the precise mathematical form of a function of several variables, we may know the value of the function at several points, and we should be able to make use of this information during a simulation. For example, a biologist may not know the precise relationship among the amounts of tryptophan repressor present, of tryptophan present, and of activated repressor present (the repressor protein becomes activated by binding tryptophan). However, the biologist may know that repressor concentration varies over a small range of values, and may know within this range, for several concentrations of tryptophan, how much activated repressor exists. In fact, biologists have determined this information experimentally for three crucial values of tryptophan concentration, and also know the approximate slope of the binding curve at these points.

A mapping frame is used to represent just this type of information for a function. A

```

Clamped.Influences:      (Total.trpR Total.trpR.normal)
Input.Variable:          trp
Output.Variable:         Activated.trpR
Points:                  ((0 0)
                          (equilibrium.trp.concentration
                           (Total.trpR * .5))
                          (trp.excess.threshold Total.trpR))
Monotonic:               T
Functional.Form:         UNKNOWN
Slope:                   INCREASING

```

Figure 3.3: A mapping that describes how the variable `Activated.trpR` is influenced by the variable `trp` when the variable `Total.trpR` is clamped to the value `Total.trpR.normal`. When `trp` is at its equilibrium concentration (a recorded value within the system), `Activated.trpR` will be one-half of `Total.trpR`.

given mapping fixes all variables but two, and lists a set of corresponding values of the free variables. A number of mappings may be used to describe different combinations of variables for one function, or different parts of the domain and range of a function. We can specify that a mapping is monotonic and/or linear, which can aid in interpolating between points on the mapping. Mappings are similar to Forbus' *correspondences* [Forbus84], but contain more information about the function they describe, and are used in additional types of inference, such as interpolation (described in Section 3.4.1).

With these representational tools, we may express a piecewise linear approximation to a function. Biologists reason about enzyme kinetics in this way. Enzyme binding curves are often S-shaped, but biologists encode and reason about them as three linear segments.

The mapping in Figure 3.3 describes, at three different points, how the amount of `Activated.trpR` varies with `trp` when `Total.trpR` is at its normal concentration.

There is a potential consistency problem that results from describing the same function by both pairwise interactions plus functions, and by mappings. Nothing prevents us from asserting that a function is linear using one representation, but that the same function is quadratic using the other representation. Future research could derive rules for checking this

consistency automatically.

DeKleer and Brown use *qualitative constraints* to represent the interactions among variables. These are abstractions of differential equations created by removal of the constant coefficients (which become irrelevant given their space of values). For example, this equation describes a sum of pressure differentials: $[dP_{in,out} + dP_{out,s} - dP_{in,s} = 0]$. Kuipers uses similar qualitative constraints, and, in addition, is able to state that one variable is a monotonically increasing or decreasing function of another variable. Iwasaki and Simon use arithmetic constraints that are similar to deKleer and Brown's.

Our relationships are unidirectional, unlike the constraint languages used by deKleer and Brown, and Iwasaki and Simon, which force us to write bidirectional relationships where they might not exist (for example, the relationship between the position of an electrical switch and the flow of current). Such bidirectional relationships give rise to the problem of causal ordering described in [Iwasaki86].

Thus, all of the representations used by other researchers are a subset of ours. Some are unable to express information in as much detail ours can (such as quantitative constraints); all are unable to represent the less precise interactions that we can represent, such as, "Y is proportional to the product of a monotonically increasing function of X_1 , plus a linear function of X_2 ."

Sacks has developed a qualitative reasoning system that uses a different approach [Sacks85,Sacks85a]. Rather than propagate values through a network of variables to derive a device's behavior, his system calls on MACSYMA to solve symbolically the set of differential equations embodied by the network. MACSYMA obtains an analytic description of the system's behavior over time. This solution is represented in a sufficiently declarative form that it can be explained and reasoned about by his system. This approach is excellent when the closed-form solution can be obtained. Our domain defies closed-form solution, however, because of the complexity of some of the interactions within it, and because some interactions are simply not

known with precision (such as mappings), and hence cannot be solved analytically.

Simulation

We are interested in simulating the behavior of the trp operon over a sequence of discrete time points. Thus, to simulate the state variable network in Figure 3.1, we first make assertions about the exogenous variables of the system, such as the initial concentrations of `Total.trpR`, `RNA-Polymerase`, and `trp`. We then derive the values of all state variables at the next point in time by propagating these initial values through the interactions described by the state-variable network.

A Framework for Predicting Device Behavior We predict device behavior using a variant of depth-first traversal of a digraph. The nodes of the graph are state variables, and edges link variables between which interactions have been defined (see Figure 3.1). The traversal algorithm maintains three sets of nodes: K (variables with known values), U (variables with unknown values), and X (the subset of the variables in U that will become known on the next iteration). Prediction is accomplished using the following procedure:

1. Initialize all exogenous state variables by creating one vin for each state variable and storing the variable's initial value in this vin. Initialize K to this set of vins.
2. Create a new set of vins for each state variable's value on the next clock cycle. Set U to this set of vins.
3. Set X to all nodes in U adjacent to some node in K .
4. For all nodes x in X do: Compute the value of x by propagating values of the state variables that influence x through the relevant interaction descriptions.
5. Set $[K = K + X]$, and $[U = U - X]$.
6. If U is not empty, go to 3.

This procedure traverses the entire graph, and is applied during each simulation clock cycle.

Three complications arise here. First, in step 3, it may be that values are not known for all the variables influencing the current x . If the value of an input variable has not yet been determined, we *backchain* through the graph, recursively attempting to determine the value of a *needed* variable from its inputs (which thus become needed variables), until we arrive at variables whose values are indeed known. This backchaining process will succeed as long as all exogenous variables (variables that are not influenced by any other variables) have values supplied for them. Variables will lack values only if the simulation was not initialized correctly.

The presence of cycles in the graph due to feedback loops in the system presents a potential problem; without some care, we might backchain around these cycles endlessly. We solve this problem by explicitly specifying that a given variable breaks a feedback loop. Backchaining must thus stop at this variable; its value is computed from the values on the previous time point of the variables that influence it. Thus, we compute the current value of `Activated.trpR` from the current value of `Total.trpR` and the *previous* value of `trp`.

The second complication is that the propagation computation in step 3 is nontrivial. It is described in the section that follows.

Other researchers have used similar approaches. DeKleer and Brown predict device behavior using a combination of techniques [deKleer84]. Normally, they propagate qualitative values from one variable to another via constraints (essentially the same as our method), using an arithmetic they defined over these values. Their procedure may reach an impasse just as ours does if values for some variables have not yet reached the current constraint. At this point they use heuristics to guess a value for a variable. If such an assumption later violates a constraint, it can be retracted and a new assumption can be introduced. Because their value space is so small, only a small number of assumptions are syntactically possible at a given point.

Kuipers does not perform simulations by propagating values from one variable to another via constraints. Instead, he has rules for generating the set of allowable values that each variable

can take on the next cycle of simulation. These generated values are then filtered according to several criteria, including the variable interaction constraints. Kuipers' propagation algorithm does not become stuck like that of deKleer and Brown. Kuipers *always* introduces new values as assumptions — the incorrect assumptions are then filtered out. We do not use this technique of introducing assumptions; when the system needs the value of a variable it simply works backward through the network to derive the value. It seems unlikely that Kuipers' technique would work with our value representations — we would have to introduce and filter a huge set of quantitative assumptions. That is, Kuipers' technique works because his state variables may take on only three possible values, so it is computationally feasible to explicitly introduce assumptions that a variable takes on all values from this set. When this set is extended to include quantitative values representable by the quantity lattice, it becomes infinitely large.

The third complication is that we wish to integrate the behavior of the system over a series of simulation clock cycles to infer its steady-state behavior in situations involving feedback loops. Does enzyme production reach an equilibrium, does it oscillate between two or more rates, or does it grow without bound? Previous researchers have tried three different approaches. First, deKleer and Brown are concerned more with *detecting* that feedback is occurring than with producing a detailed prediction of the result of the feedback, such as predicting the approximate point at which negative feedback stabilizes. Their coarse representation often gives rise to multiple, underdetermined types of feedback, which are reported to the user.

Second, Weld's aggregation technique makes inductive inferences about the eventual behavior of the system by extrapolating from one or more short-term states of the system [Weld86].

Third, Iwasaki and Simon [Iwasaki86] have applied the method of comparative statics to the analysis of feedback. This approach requires the user to state second-order differential equations that describe explicitly how the system changes over time, and to solve these equations symbolically. This approach appears to be promising, although it assumes that these second-order equations can in fact be stated and solved. Much work remains to be done in

determining the strengths and weaknesses of these approaches.

Propagation The second complication discussed in the previous section results from the rich value and interaction representations we employ. The propagation step requires us to evaluate an interaction description based on the values of its input variables (analogous to evaluating an algebraic expression). The problem is that we have several types of value representations, and several types of interaction descriptions, and thus, quite a few different potential types of propagations to be performed. For example, in one case we might have to propagate the value $[trp > trp.minimal]$ through the expression $[activated.trpR = trp * trpR]$; in another case, we might have to propagate a quantitative value for trp through a mapping, or through an interaction that is known to be only linear. In contrast, deKleer and Brown have a 3×3 lookup table describing how to propagate values through the operations of multiplication, division, addition and subtraction — a much simpler process.

We solve this propagation problem by identifying several important subclasses of the problem, and defining procedures for solving these subclasses. Given that there are a number of combinations of a type of value to be propagated through a type of interaction, we have partitioned this set into a number of solvable instances. (There are cases, however, where the value of the output variable is completely unconstrained.) Whenever a propagation step is to be performed, all the inference classes are considered, since the information provided by one class will not necessarily be a subset of the information provided by another class. All classes are implemented as LISP procedures that query the quantity lattice to test the preconditions for the rule of inference; if the rule succeeds, it stores the propagated value in the quantity lattice. We now describe these classes and their associated rules.

Class 1: Quantitative Calculations

In this case, the interaction in question is a quantitative arithmetic expression, such as $[A = 2B + C]$, and exact quantitative values are known for all the variables to be propagated.

We can perform a quantitative calculation; we compute the value of A by doubling the value of B , then adding the value of C .

Class 2: Use of Mappings

In this case we attempt to evaluate a function by evaluating mappings that are associated with it. First, the system must determine that the mapping applies at all — that the fixed variables for this mapping have their required values (in Figure 3.1, it must be the case that $[Total.trpR = Total.trpR.normal]$). Next, if the current value of the input variable recorded is part of a recorded point in the mapping, we can set the output variable to the associated value from the mapping. So, if $[trp = equilibrium.trp.concentration]$ then we can assert that $[Activated.trpR = Total.trpR * .5]$.

Class 3: Interpolation of Mappings

A more complicated situation arises when a mapping point does not exist for the current value of the input variable. Using the mapping in Figure 3.3, we can compute $Activated.trpR$ when $[trp = equilibrium.trp.concentration]$. But what if we know that $[trp = 4 * equilibrium.trp.concentration]$? If we know that the mapping is monotonic, we can infer that $[Activated.trpR > .5 * Total.trpR]$. And if we also know that this mapping represents a linear function with an intercept of 0, we can further infer that $[Activated.trpR = 2 * equilibrium.trp.concentration]$ (quadrupling the input quadruples the output).

Class 4: Relative Calculations

Biologists often wish to compare the behaviors of a system under varying circumstances. For example, if the behavior of the system is known under a given concentration of tryptophan trp_1 , it may be desirable to predict the behavior at the concentration $[trp_2 = trp_1 * 2]$. In the previous paragraph, we interpolated a linear mapping. If we have a number of interactions whose precise equations are not known, but which are known to be linear with a y -axis intercept of 0, we can compute values for variables in the new simulation run that are multiples of 2 (in this example) times the values in the old prediction. This class of inferences is similar to

class 3, but it uses only information about the form of the equation, whereas class 3 also uses mapping information.

Class 5: Qualitative Calculations

When the information available does not allow us to use any of classes 1-4, we attempt to apply the following inference rules to constrain the value of the output variable. For example, if we were attempting to evaluate an interaction that was a product of two variables, we could try to employ rule 4 in the following list. These rules are also used in Simmons' quantity lattice [Simmons86]. In these rules, *rel* is one of $\{<, \leq, =, \neq, \geq, >\}$:

1. $(X \text{ rel } Y) \supset (X - Y \text{ rel } 0)$
2. $(Y \text{ rel } 0) \supset (X + Y \text{ rel } X)$
3. $(Y \text{ rel } 0) \supset (X \text{ rel } X - Y)$
4. $((X \text{ rel } 0) \wedge (Y > 0) \wedge (Y < 1)) \supset (X \text{ rel } X * Y)$
5. $((X \text{ rel } 0) \wedge (Y > 1)) \supset (X * Y \text{ rel } X)$
6. $((X > 0) \wedge (Y > 1)) \supset (X * Y > X)$
7. $((X < 0) \wedge (Y > 1)) \supset (X * Y < X)$
8. $((X > 0) \wedge (Y < 0)) \supset (X > X * Y)$
9. $((X < 0) \wedge (Y < 0)) \supset (X < X * Y)$

For example, rule 5 states that, if the program is attempting to evaluate the product of X and Y when X is known to be positive and Y is known to be greater than 1, then it can infer that their product is greater than X .

Class 6: Monotonicity Calculations

If we are evaluating a function that is known to be monotonic, at least over the relevant region of its domain, then analysis of its inputs may allow us to deduce whether the function

increases, decreases, or remains constant with respect to its value at a previous time point. If all inputs remain constant or we can prove that the changing inputs cancel one another, then the function remains constant. Also, if the signs of all changing inputs are the same, then the function changes in the same direction as its inputs. For example, if in computing `Total.trpR` we can prove that the current value of `Free.trpR` is unchanged from a previous value, and if the current value of `trp` has increased over a previous value, then we can deduce that the current value of `Total.trpR` is greater than the previous value. Other researchers have recognized this property of monotonic functions, but have usually applied it in a much simpler context, without using such diverse information about values and interactions, or about value histories.

Consider how this approach compares to that of deKleer and Brown. Because their representations do not encode values very precisely, their program is often unable to resolve competing influences. For example, the value of the sum $[dA + dB]$ is undefined if $[dA = -]$ and $[dB = +]$. At these times, they generate branching predictions (called *envisionments* to model all possible behaviors of the device. Envisionment is an interesting and important ability, but experts are often able to resolve the ambiguity. When our system reaches such a condition, it instructs us to add more information to its knowledge base or to the problem statement, which our representation allows us to do.

3.4.2 An Example Model 2 Prediction

One important component of our simulation environment is a library of landmark values for variables: values of interest for the variables that are permanently stored in the system's quantity lattice. They might be of interest because they are extreme values, or because they are mentioned in mappings. The library vins that follow, for example, describe three different transcription-initiation rates that are related by multiplicative constants. When we describe a vin, we give its name followed by knowledge the system has about how it is related to other

vins. Library vin names are often of the form `Variable.description`, and simulation value names are usually of the form `Variable.simulation-cycle`. Sample names are `trp.excess` and `trp.1`, respectively (the latter describes the value of `trp` on the first simulation cycle).

`Transcription.Initiation.Rate.minimal`

`(= (* Transcription.Initiation.Rate.maximal .0125)`

`(= (* Transcription.Initiation.Rate.equilibrium .066667)`

`(> 0)`

`Transcription.Initiation.Rate.maximal`

`(= (* Transcription.Initiation.Rate.minimal 80))`

`(= (* Transcription.Initiation.Rate.equilibrium 5.33))`

`Transcription.Initiation.Rate.equilibrium`

`(= (* Transcription.Initiation.Rate.minimal 15))`

`(= (* Transcription.Initiation.Rate.maximal .1875))`

In Figure 3.4 and Figure 3.5 we show what information has been derived about the value of every state variable on two successive cycles of a simulation. In the first cycle, the system has been told that $[trp.0 = 0]$, and it predicts that the bacterium begins to synthesize the `trp` enzymes and `trp` at a high rate. On the second cycle, the enzymes are synthesized at a lower rate, since `trp` begins to inhibit transcription to a small degree, but `trp` is synthesized at a higher rate because more enzyme has accumulated. A number of mappings are used to compute the values of various interactions relative to values in the library.

Information about values in cycle 2 is recorded relative to values in cycle 1, so we know, for example, that the `Transcription.Initiation.Rate` has decreased. Also in cycle 2, the system computes a value for `trp-mRNA.2` by combining the new mRNA synthesized in that cycle, with the mRNA that has been degraded during that cycle. A mapping tells the system



Figure 3.4: A prediction of the values of system state-variables after the first tick of the simulation clock. Each variable is annotated with the constraints on the value of the variable that are stored in the quantity lattice.

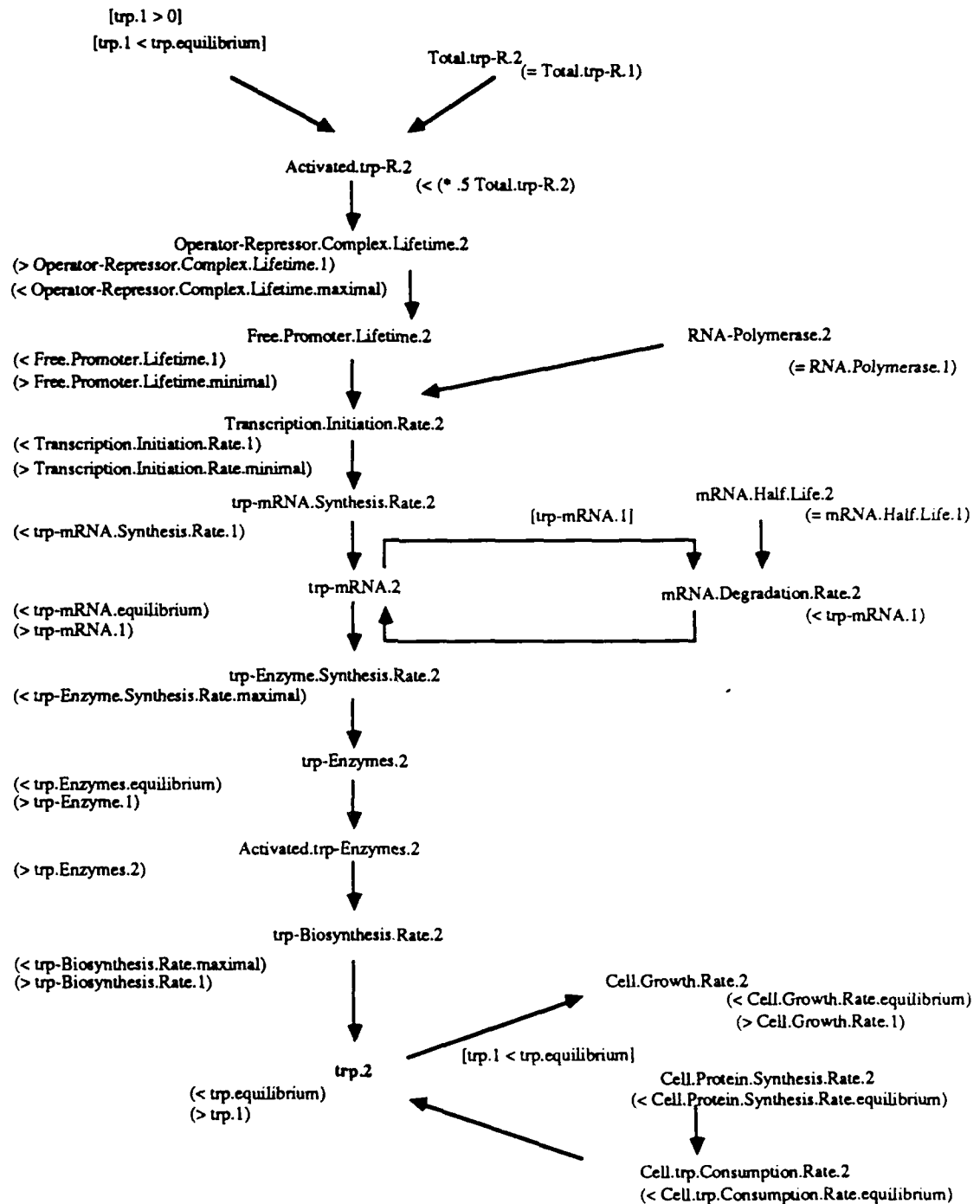


Figure 3.5: A prediction of the values of system state-variables after the second tick of the simulation clock.

that, since the concentration of mRNA is below its equilibrium value, synthesis is a stronger effect than degradation.

3.4.3 Analysis of Model 2

Traditional numerical simulation methods cannot predict the behavior of a system if they are not provided with quantitative initial values for the system state-variables, plus precise mathematical descriptions of the functional relationships among these variables. Existing qualitative reasoning techniques relax these requirements because they can manipulate very imprecise values for system state-variables, and can accept descriptions of functional relationships that have been abstracted to a degree. However, these techniques produce very imprecise predictions of the behavior of the system. The techniques described in this section do not require quantitative initial values, but they can produce precise predictions if such values are available. Our methods also do not require precise descriptions of the functional relationships among state variables, and they provide a language for describing functional relationships with varying degrees of precision.

Several limitations exist for these methods. First, the system contains no explicit representation of the rate of the simulation clock. Thus, it is unable to determine the absolute total output of a process whose output accumulates over time. It should be possible, however, to use the representations we have developed to make assertions about process rates that would let us infer that, for example, after 10 clock intervals of maximum synthesis the trp enzymes would accumulate to their equilibrium levels. Second, it is possible to provide the system with too little information to make predictions with some desired degree of precision. We have yet to determine the exact relationship between the precision with which state-variable initial values and interactions are described, and the expected precision of the simulation.

Finally, this approach to modeling the trp system has turned out to be too inflexible for our needs because of the static description of state variables and of their interactions. The

model we have developed is valid for only a certain experimental configuration. If we wish to simulate a different experiment — for example, with a mutant bacterium growing in a different medium — a somewhat different set of state variables and interactions will be required (since different media activate different enzymatic pathways). A more practical approach would be to specify what objects are present in a given experiment, and to let the system derive the state-variable network dynamically from its knowledge of possible interactions among classes of objects, such as the interactions between an enzyme and the chemicals with which it reacts. It would also be possible to have the system always reason with a reaction network containing all known reactions, but ignore reactions for which any inputs were not present.

3.4.4 Summary of Model 2

In a complicated, natural domain such as regulatory genetics, it is rare that theories are described with complete, quantitative precision. Existing types of qualitative descriptions alone, however, do not suffice to capture the knowledge that scientists have about the tryptophan operon. Therefore, we have been motivated to develop methodologies by which a range of types of knowledge, from qualitative to quantitative, can be described and jointly used to simulate device behavior. These methodologies include representations for state-variable values and for the mathematical interactions between state variables, and reasoning processes for propagating state-variable values through these interactions.

We have synthesized and extended the prior work of several other researchers in this field. Our “devices” — bacterial operons, regulatory proteins, and the like — are more complicated and less well understood than are the devices that some of our colleagues have attempted to model, but we desire more precise predictions of their behavior than other researchers have accepted for their devices.

3.5 Model 3: The Process Model (GENSIM)

The third model of the trp system that I developed is called GENSIM (*genetics simulator*). It is based on a combination of ideas from the previous two models and from Forbus [Forbus84]. As in model 1, GENSIM addresses the problems of describing the initial conditions of an experiment, representing a theory of chemical reactions, and using that theory to predict what reactions occur in a given experiment and what the results of those reactions will be. GENSIM describes a gene-regulation experiment by specifying what objects are present at the start of the experiment and what their properties and relationships are. These objects are represented as frames in a KEE knowledge base (KB). Each object is an instance of a general class of biological objects; these classes of objects are defined in a second KB.

A third KB describes *processes* that GENSIM uses to detect chemical reactions among the *objects that exist in an experiment*. These reactions can result in the creation of new objects, and are used to predict future states of the gene-regulation system. GENSIM processes create objects and manipulate the latter's properties, but do not reason about quantitative state variables such as concentrations. This type of reasoning was explored in model 2, but has not been integrated into GENSIM. Thus, GENSIM predicts *what* objects are produced in an experiment, but not the *amounts* of the objects that are produced.

GENSIM embodies a qualitative chemistry whose features are as follows:

- Chemical objects represent populations of molecules
- The decomposition of objects into their component parts is represented
- Chemical processes are represented as frames and are arranged in an inheritance hierarchy; many processes inherit portions of their definitions from more general process classes
- Chemical reactions are probabilistic events that act on populations of molecules

- Because of their probabilistic nature, chemical reactions split each reacting population of molecules into two subpopulations: those that do and those that do not react
- In order to ensure that correct simulations are computed, the syntax of reaction preconditions must be restricted. This restriction is a requirement of the simulation algorithm, but has a valid chemical interpretation.

The section begins by describing a part of the trp system that will be used as an example later in the section. It then discusses the class KB and the representation of object part-whole structures. Next, it describes the process KB. The following subsection provides an overview of the extent of the biological knowledge contained by GENSIM. Next is a description of the algorithms used by the simulation program that computes predictions of experimental outcomes for GENSIM.

3.5.1 An Example

The mechanism of transcription will be used as an example throughout the remainder of this chapter. Transcription is a set of processes that are involved in the expression of the genes within the trp operon. Transcription is somewhat analogous to copying a magnetic tape. An enzyme (called *RNA polymerase*) first attaches to the trp operon DNA at a site called a *promoter*, and then moves along the linear DNA strand, reading the message on the DNA and simultaneously synthesizing another long molecule called RNA that contains a copy of the DNA message. When RNA polymerase recognizes a *terminator* DNA site, it releases both the DNA and RNA (see section 2.1.2 for more details).

3.5.2 Class Knowledge Base

GENSIM's *class knowledge base* (CKB) is a taxonomic hierarchy of classes of biological objects such as genes, proteins, and chemical binding sites. The KB describes the properties and states of different classes of objects, such as the decomposition of objects into their component parts.

The CKB can be viewed as a library that records all types of objects that could be present in experiments on the trp system (in practice, we have omitted many largely irrelevant objects because bacteria are incredibly complex biochemical systems). Each class is represented as a KEE class frame (or *unit*). The CKB is shown in Appendix A.

Modeling of a specific biological experiment begins with a specification of what actual objects (as opposed to classes) are present in the experiment. These objects are represented as KEE instance units that are defined as children of the appropriate CKB class. Instances inherit many of their properties from the class. For example, the CKB describes the class of enzymes called *RNA-Polymerase*, which is instantiated to an instance such as *RNA-Polymerase.1* within a simulation. The term *simulation knowledge base*, or *SKB*, refers to a KB containing the KEE instance units that represent the objects present in an experiment of interest. The SKB corresponds to the working memory of a production system. Since the facts it contains are represented using units, all facts are literals and contain neither disjunction nor negation.

In the ontology for this qualitative chemistry, each "object" represents not a single molecule, but rather a homogeneous *population* of molecules. For example, all molecules of tryptophan in a experiment that are in a given state (such as floating free in solution) are represented by a single KEE unit. All molecules of tryptophan in a different single state (such as bound to the trp-repressor protein) are represented by a different single KEE unit.

A central type of relationship among objects in this domain is the *containment* of one or more objects by a composite object. The example object structure in Figure 3.6 describes an experiment class that has two components: an enzyme (*RNA-Polymerase*) and a segment of DNA (*Trp.Operon*). The DNA is in turn divided into a number of component regions. Several issues of interest arose in the representation of objects with complex component structures: how to represent different types of containment relationships, how to define classes of such objects, and how to instantiate a class for such an object. The following slots within the *Trp.Operon* class are used to represent the component structure of this object (each slot is

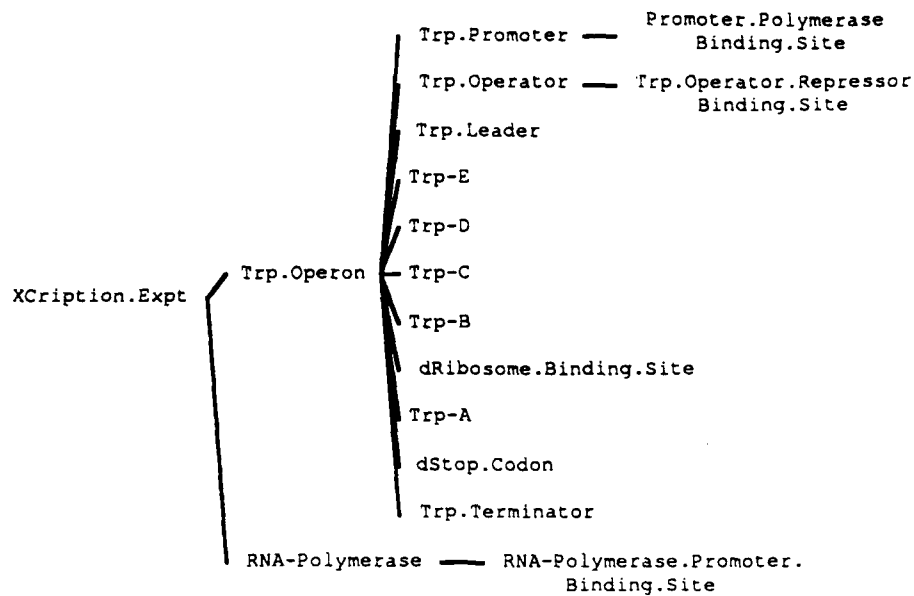


Figure 3.6: The objects in a transcription experiment. This experiment contains the trp operon and RNA polymerase, both of which have the internal structure shown.

truncated for brevity):

Trp.Operon

```

Component.Object.Classes:  Trp.Promoter Trp.Promoter Trp.Operator Trp.Leader
                           dRibosome.Binding.Site Trp-E dStop.Codon ...

Component.Object.Bindings: $pro1 $pro2 $op1 $lead1 $drbs1 $trpe $dsc1 ...

Component.Objects:

Relation.To.Components:   IRBound IRBound IRBound IRBound IRBound ...

Structural.Relations:     (PUT.VALUE $Object 'UnRegulated.Promoters $pro1)
                           (PUT.VALUE $pro2 'Regulator $op1)
  
```

A user instantiates `Trp.Operon` by sending a `Create.Instance` message to the `Trp.Operon` unit. The LISP method invoked instantiates `Trp.Operon` and then recurses, sending the same message to each of the class units listed in the `Component.Object.Classes` slot of `Trp.Operon`. Instantiating `Trp.Operon` itself involves creating a new instance of the class with a unique name, such as `Trp.Operon.1`. The names of the created component objects are recorded

in the `Component.Objects` slot of the `Trp.Operon.1` object, and are bound to the variables named in the `Component.Object.Bindings` slot of `Trp.Operon`.

There are some objects in the `trp` system that contain component objects whose slots must refer to one another. For example, every promoter object in an operon records what operator object in that operon control the promoter. We wish to specify this general relationship in the CKB, but whenever a user instantiates such an operon, the promoter within that operon should refer to the operator object within that operon object, and not to operators within other instances of the operon. GENSIM allows the user to list a set of variablized assertions within the `Structural.Relations` slot of an object class. These assertions are executed by `Create.Instance` using the variable bindings described in the previous paragraph.

`Trp.Operon` is a *dissolved* component of the experiment class, whereas `Trp.Promoter` is *irreversibly bound* to `Trp.Operon`. These different relations between container and component are important distinctions chemically: dissolved components are floating free in a containing solution, whereas irreversibly bound components are chemically bonded to their containers. The `Relation.To.Components` slot describes the nature of the component relationship between `Trp.Operon` and each of its components.

GENSIM does not represent temporal aspects of objects explicitly; the work of [SimmonsM87] and [Williams86] is relevant to this problem. GENSIM's task is to simulate the behavior of a chemical system during a very short interval of time. Within such a short interval, new objects can come into existence because the creation of an arbitrarily small amount of an object is enough to change its concentration from zero to positive. However, we make the simplifying assumption that GENSIM simulations take place in a short enough interval that a population of molecules is never fully consumed; thus objects are never deleted from simulations. When an arbitrarily small amount of an object is destroyed, we cannot assume that its concentration has changed from positive to zero.

This assumption implies that the number of objects in a simulation must increase monotonically. In reality, biologists do perform experiments over intervals of time long enough that objects are completely consumed by reactions. However, this assumption simplifies the implementation of GENSIM significantly — without it, GENSIM would have to reason about time and quantities. Yet the system is still able to make predictions for an interesting class of experiments. This assumption had an interesting and unexpected side-effect: It simplified the implementation of HYPGENE in several ways, discussed in Chapter 5.

3.5.3 Process Knowledge Base

The *process knowledge base* describes the behaviors of the objects in the trp system. For example, processes describe chemical binding, rearrangement, and dissociation events.

The notion of process used in GENSIM is similar to that used by Forbus [Forbus84]. GENSIM processes are most similar to what Forbus terms *individual views*, because both are concerned with the creating and deleting objects, and altering relations between objects. Forbus uses the term *process* to refer to entities that describe how quantities change in a physical system. Other differences in our approaches will be noted where appropriate.

Processes are described as KEE units. The sample process on page 288 of Appendix A specifies a binding reaction between the activated trp-repressor and the trp operator. Processes are executed by a *process interpreter*. The precise meanings of these slots will be explained in the next section; briefly, processes specify actions that will be taken (listed in the **Effects** slot of the process) if certain conditions hold (listed in the **Preconditions** and **Quantity Conditions** slots). In addition, since processes operate on objects, the **Parameter.Object.Classes** slot specifies on what types of objects a process acts. The interpreter activates a process when at least one object is present from each class in this list. The **Parameter.Objects** slot lists variables that are bound to the actual objects with which a process has been activated. In addition, an arbitrary list of variable bindings may be given in the **Bindings** slot. The process

Predicate or Function	Meaning
(OBJECT.EXISTS X Y)	Object X exists within class Y
(IS.PART X Y)	Object X is part of object Y
(MEMB X Y)	Atom X is an element of list Y
(GET.VALUES X Y)	The value of slot X of object Y
(BINDV \$X Y)	Binds variable \$X to the value Y
(CREATE.COMPLEX X Y)	Creates an object of type X containing the objects in list Y as parts
(COPY.STRUCTURE X)	Creates a copy of object X
(PUT.VALUE X Y Z)	Stores Z into slot X of object Y

Table 3.1: GENSIM predicates and functions. OBJECT.EXISTS, IS.PART, and MEMB are predicates. The symbols AND, OR, NOT, EXISTS, and FORALL have their standard logical meanings.

on page 288 in Appendix A specifies that for any two molecules of type `Trp.Operator` and `Trp-Repressor`, if these objects contain binding sites that are specific to each other, and if these binding sites are empty and contain no mutations that interfere with this binding reaction, then new instances of these objects should be created and bound together as a new object. Table 3.1 explains the functions and predicates used within GENSIM processes.

Processes possess an additional type of precondition called `Efficiency.Preconditions`. These preconditions prevent process invocations that, although technically correct, are uninteresting. For example, the `trp-repressor` protein binds to the operator site at the front of the `trp` operon. It can bind there during almost any of the 17 intermediate steps of the transcription process. These intermediate complexes, however, have no special functionality when bound to the repressor, and are thus uninteresting. Large numbers of such process activations are prevented through use of the `Efficiency.Preconditions`. We might argue that these useful but technically incorrect process preconditions could confuse the HYPGENE system. I store these preconditions in a separate slot precisely so that HYPGENE can ignore them.

The library of processes is structured as an inheritance hierarchy, part of which is shown

in Appendix B. At the top level is a general template for all processes. Its children are templates that provide general descriptions of different classes of events, such as chemical-binding and enzymatic-reaction processes. In turn, the children of these templates either describe actual processes (such as the bindings of particular species of molecules), or define important subclasses of processes. An example of such a subclass is **Mutually.Exclusive.Binding**. This template defines preconditions for a subclass of binding processes such that, for object A to bind to B, it cannot be the case that A is already bound to an object of class B or that B is already bound to an object of class A. A particular process instance can inherit slots from one or more template processes. For example, **Trp-Repressor.Binds.Trp** inherits most of its definition from the **Repressor.Binds.Cofactor** template, which specifies the behavior of a general class of proteins. However, it inherits additional preconditions from the **Mutually.Exclusive.Binding** process. Process classes often provide enough of the definition of a process instance that only the **Parameter.Object.Classes** slot must be modified in the instance — to define the actual classes of objects to which the process pertains. It is extremely easy for a user to define new processes that bear this much similarity to their parents.

Additional machinery is provided to facilitate the use of inheritance to modify process templates. Pairs of slots are actually provided for each of the process preconditions, bindings, and effects — called for example, **Effects.M** and **Effects.A**. The values of **Effects.M** (*main*) slot are inherited using KEE's *override* inheritance; thus, new values for this slot override previous value(s). Values of **Effects.A** (*additional*) are inherited using *union* inheritance; new values for this slot are added to previous value(s). The process interpreter executes the effects listed in both of these slots. The pairs of slots for preconditions and bindings are defined analogously.

This use of inheritance to define processes is similar to that used in object-oriented programming (OOP) systems [Goldberg83, Stefik86] (although GENSIM's process interpreter does not employ the message-sending control structure of OOP). Thus, our approach reaps many

of the same benefits as OOP systems, such as facilitating the definition of new processes and the maintenance of existing processes. GENSIM's use of inheritance is novel in two ways. First is the use of inheritance in this particular type of language — a production-rule-like process-description language. Because GENSIM processes are so similar to the production rules used in expert systems, inheritance probably could be used as a software-engineering tool in the construction of expert systems. The second novel aspect of this use of inheritance is the manner in which an individual process is dissected into pieces (preconditions, effects, parameters, objects, etc) that can be inherited and modified as required. OOP systems usually treat procedures as indivisible units that can only have additional code wrapped around them, rather than having their internals altered as desired.

The definition of a process inheritance hierarchy should have benefits in addition to those already defined. This approach should facilitate the definition of new processes by HYPGENE: HYPGENE can create new processes by instantiating process templates (see Section 5.8.3 for more details).

Forbus [Forbus84,Forbus86] and Simmons and Mohammed [SimmonsM87] use a similar notion of process in their process-modeling systems; both include parameters, preconditions, and effects. Forbus briefly discusses the use of an abstraction hierarchy to define processes [Forbus84] (p.44). The main difference in our approaches is that Forbus requires that, for a process P_1 to be a specialization of another process P_2 , the parameter-object classes, preconditions, and quantity conditions of P_1 must be a subset of those of P_2 . Since our `Preconditions.M` slot allows parent preconditions to be removed in a child process, we do not use this restriction. In addition, Forbus does not use inheritance from multiple parents. There are also differences between our process-definition languages. The preconditions of Forbus' processes must contain a conjunctive list of (possibly negated) predicates; the preconditions of GENSIM processes can include arbitrary predicate-calculus formulae, including disjunction and quantification. In addition, GENSIM and the system built by Simmons and Mohammed allow process effects to

be conditionalized and universally quantified; Forbus does not allow this. Forbus' framework for evaluating processes is similar to ours in that Forbus first computes "the process and view structures" and then "resolves influences."

3.5.4 Representational Tradeoffs

At least two other approaches could be used to represent object behaviors. One approach would eschew any use of processes, and use slots within each object to represent the behaviors of the object. For example, for an object *O*, one slot might list all objects that *O* can react with, and other slots might list other objects that, when part of *O*, inhibit or activate these reactions. Another slot might list the products of the reactions. Several problems arise here. First, a given object might participate in several reactions, all of which could involve different other reactants, and produce different products. Thus, some encoding of slot values must be used to segregate the different reactions. Second, if five objects participate in a given reaction, each object must describe the same reactants, products, activators, inhibitors, and so on, which is redundant. Third, as GENSIM processes illustrate, it usually is not enough simply to list the activators and inhibitors of a reaction; we usually must test for particular properties of these objects using complicated predicate-calculus formulae.

A second approach would use processes that are somewhat more general than those currently used by GENSIM. To model three different repressible operons using GENSIM — say, the *trp*, *lac*, and *phe* operons — we must create separate GENSIM processes to describe the binding of the *trp*-repressor to the *trp* operator, the *lac*-repressor to the *lac* operator, and the *phe*-repressor to the *phe* operator. GENSIM allows these processes to be constructed using inheritance from the general `Repressor.Binds.Operator` process but we still might argue that this approach creates an excessive number of processes. The alternative would be to use only a general process, such as `Repressor.Binds.Operator`, and to let each repressor object specify

the operator(s) to which it can bind. Although the latter approach is considerably more compact, it has two disadvantages. It forces a proliferation of the slots that encode specificities — every general process (such as `Repressor.Binds.Operator`) would employ such a slot (in fact, GENSIM's representation of mutations follows this model). Second, this approach assumes that all repressor-operator binding reactions can in fact be described by a single general process. Given the diversity of biological systems, this assumption is likely to be false. For example, a certain repressor protein might contain more than one operator-binding site, and thus require special preconditions and effects. Of course, the behavior of this protein could be modeled with a different general process.

3.5.5 An Assessment of GENSIM's Knowledge

This section provides a description of the breadth and depth of the biological knowledge present in GENSIM to give the reader an understanding of the complexity of this program and of its knowledge. The examples in Chapter 7 should also help the reader to assess GENSIM's abilities.

Appendix A contains the CKB; the PKB is presented in Appendix B. Both knowledge bases are taxonomic hierarchies that were constructed using the KEE frame-representation system. The CKB contains definitions of general classes of biological objects, such as enzymes, operons, promoters, amino acids, and active sites within proteins. It also contains knowledge of all the specific objects within the *trp* system, such as each enzyme within the *trp* biosynthetic pathway, the *trp* operon, each specific amino acid, and each gene within the *trp* operon. Although the internal structures of many objects are described at a gross level (for example, the *trp* aporepressor protein contains two binding site objects as components), a significant amount of structural knowledge has been omitted from the model. GENSIM contains neither DNA nor protein-sequence data, nor representations of the three-dimensional structures of proteins, nor knowledge of chemical structure. However, biologists obtained this type of morphological knowledge only in the 1970s — after many aspects of attenuation were known. Thus, GENSIM

contains knowledge of virtually all the objects within the trp system as they were known to biologists in the 1960s.

GENSIM lacks knowledge of biological systems other than the *E. coli* trp operon, such as other gene-regulation systems and enzymatic pathways. Although hundreds or thousands of such systems are known today, relatively few were known in the 1960s; furthermore, such knowledge is of only marginal relevance to work on the trp operon. Yet, this type of knowledge was sometimes relevant; examples include the work in the *E. coli* phe and his operons, and in the trp operons of other bacteria.

The PKB also contains both general knowledge (of general classes of reactions, such as the general pattern that repressor proteins bind operator regions of DNA), and specific knowledge (of reactions in the trp system, such as the reactions in the trp biosynthetic pathway, and the events involved in transcription and translation). These events are represented in significant detail, but I did omit some details, such as the many chemical factors involved in transcription and translation, and knowledge of reaction mechanisms (which describe intermediate events in chemical reactions). As in the CKB, there is no knowledge of reactions in other gene regulation or other biochemical systems.

3.5.6 The Process Interpreter

GENSIM processes bear significant similarity to production rules. In like manner, the program that interprets processes is similar to a production system. This interpreter uses processes to detect interactions among objects that exist in the current simulation, and computes the effects of these interactions. This section describes how the process interpreter activates and executes processes, and how it manages the existence of objects during a simulation. Because these issues are so closely intertwined, this section alternates between the two. It begins with a brief description of object management, then presents a detailed description of process execution. It finishes by presenting a number of points related to both issues.

Before proceeding, let us resolve the potential ambiguity of the term *process instance*. We use this term to mean the execution of a process on a given set of objects — an instance of process execution. Because processes are defined hierarchically in a KEE KB, the KB contains both class process units and instance process units; we use the term *leaf process* to refer to instance process units.

Object Management

At least two possible approaches to the management of objects are conceivable in GENSIM. The term *object management* refers to the manner in which operations on objects (such as creation, deletion, and modification) are implemented. Consider the simple reactions in Figure 3.7, in which an object A is acted on by two processes: one converts A to object B, the other converts A to object C. A simulator might model the first reaction in one of two ways: it might modify A directly to produce B, or it might copy A to a new object, and then modify that new object to produce B.

GENSIM should produce correct and complete predictions: it must predict all possible interactions between objects, and only correct interactions. Predictions should not depend on the order in which processes are executed; for example, care must be taken that the execution of the first reaction in Figure 3.7 does not prevent execution of the other by removing the A object, because both reactions would occur to some degree. In general, most chemical processes are probabilistic events that act on populations of molecules. Since objects A, B, and C represent populations of molecules, and reactions occur at some finite *rate*, it is likely that, at some time, members of all populations exist. Thus, when a chemical reaction converts A to B, all members of population A do not disappear instantaneously.

We conclude that GENSIM should not destroy objects or modify their properties, because to do so would allow the possibility that the system would overlook some important object behavior. Rather than modifying A directly to create B, GENSIM copies A and modifies the

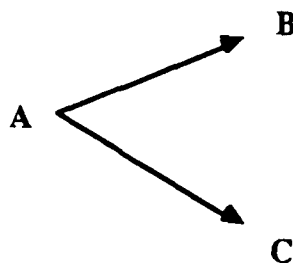


Figure 3.7: A simple reaction in which an object A can cause two reactions; one produces B and the other produces C.

copy to produce B. We term this *object forking*. The assumption that object populations are never fully consumed was discussed in Section 3.5.2.

This discussion also implies that, to predict both *what* reactions occur, and at what *rates*, requires a two-stage computation (note that GENSIM currently performs only the first stage):

1. Determine the complete set of reactions that will occur — that is, the complete set of objects that will be created and the set of processes that will fire. Forbus calls this task computing the “process and view structures” [Forbus86].
2. Use information about reaction rates and molecular concentrations to compute *how much* of each object population forms. Forbus calls this task “resolving influences” [Forbus86].

The intuition here is that to predict the rate of a reaction R that consumes reactants I_1, \dots, I_n , we must know about all other reactions in which each I_i takes part, so that we can compute the relative rates of these competing reactions.

Other researchers do not use object forking. One reason for this may be that in their domains, objects represent not populations of molecules that probabilistically change to another state, but rather individual objects that change completely. Simmons and Mohammed [SimmonsM87] use an explicit model of time to represent object properties as histories, so in effect, different versions of different objects exist at different points in time due to the different values their properties take on at these different points in time. Our domain would not allow this approach because the populations of molecules we model often may *coexist* at a single time.

My experiences with Model 1 showed the need for object forking. They also showed that some production systems are not well suited to simulations in which object forking is *not* performed. The KEE forward chainer allows a rule R to fire on a frame O only once within an execution of the forward chainer. Thus, if O is modified to produce a new frame with the same name but different slot values, the production system will not consider firing R again on O , even if the preconditions of R failed on the first O , but would succeed on the new O . This is presumably an optimization, but it has a deleterious effect for this type of simulation.

Reference Patching One complication that arises during object forking is that the component objects of a complex object may refer to one another, and these references must be altered during the copy. This procedure is called *reference patching*. In the earlier example of the `Trp.Operon` object, we noted that object `Promoter.1` contained a slot `Regulator` with value `Operator.1`. This slot is used to indicate what promoter object is affected by the state of this operator object. The value of this slot must be altered in the object to which `Promoter.1` is copied, to name the object to which `Operator.1` is copied, since a promoter is regulated by only the operator in the same operon object.

Another complication is that there are some object slots for which reference patching should not be performed, such as slots that do not refer to other objects. Thus, each slot in the system is described by a special unit — its slot descriptor — that describes whether or not reference patching should be performed for that slot.

The Process-Execution Cycle

I constructed two different implementations of the process interpreter. The first uses a brute force algorithm and is slow. It iterates through all processes in the PKB, searches for sets of objects that satisfy the preconditions of the process, and executes the effects of such processes, until no new reactions are detected. More precisely, the first interpreter cycles through the

following steps until no new processes can be activated in step 2.

1. **Process selection:** Select a member P from the set of all leaf processes that exist in the process knowledge base.
2. **Process activation (binding of processes to objects):** Consider each object class C_i ($i = 1..N$) listed in the `Parameter.Object.Classes` of P . Let O_i be the set of object instances within class C_i . If every set O_i is nonempty, then enumerate sets that contain a single member from every set O_i . This set, A , is the set of possible bindings of the `Parameter.Object` variables of P , and is a list of all possible interactions of objects due to P . If O_i was empty, no activations of this process are generated.
3. **Filter process activations:** Remove from A every set of variable bindings for which process P has been activated with that set of bindings previously in this simulation. Note that because objects never change, and process preconditions can refer only to objects in the parameter objects of a process, that the truth or falsity of the preconditions of a process instance will never change. Thus, it is never necessary to activate a process more than once.
4. **Process execution:** For each A_j in A :
 - (a) Bind the variables in the `Parameter.Objects` slot of P to the objects in A_j
 - (b) Evaluate the variable bindings in the `Bindings` slot of P
 - (c) Evaluate the `Quantity.Conditions` and `Preconditions` of P . IF any are false,
THEN CONTINUE to the next A_j ; ELSE
 - (d) Execute the `Effects` of P

This approach is inefficient in two ways:

1. It repeatedly examines every process, even if no objects exist in some parameter-object class of a process

2. It repeatedly generates process activations that have been considered previously, thus reevaluating failing process preconditions

The cost of these inefficiencies grows as more objects and processes exist in a simulation.

To improve what is essentially a generate-and-test algorithm, we move part of the test inside the generator. New process activations are generated only when the interpreter starts running, and when new objects are created. A given set of bindings is never generated more than once for a process.

The second algorithm maintains two data structures. The first is the *process-activation queue*, which lists all valid variable bindings for which each process has not yet been executed. The interpreter repeatedly pops variable-binding lists off this queue and activates the associated process with these bindings.

The second data structure is used to determine what process activations should be created when a new object is created. It is called the *live-objects* structure and consists of a list of records of the form

$$(C (P_1 \dots P_n) (O_1 \dots O_m))$$

where

- C is the name of a class of objects that appears in the `Parameter.Object.Classes` slot of at least one process
- (P_1, \dots, P_n) is the set of processes that contain C in their `Parameter.Object.Classes` slot — the processes that describe reactions involving this class of object
- $(O_1 \dots O_m)$ is the list of objects within class C that exist in the simulation

When a new object O is created by the execution of a process, the following actions are taken:

1. Let C be the class of O .

2. Find all records R in the live-objects list such that record R_i describes a class that is equal to or a superclass of C . If none exist, exit.
3. Add O to the object list of each record in R .
4. For each R_i in R do
 - (a) For each process P_j in R_i do
 - i. Compute the new variable bindings for P_j : Imagine that P_j operates on two objects — one of class C , the other of class D . The activations of P_j consist of O paired with every object from class D (as listed in the live-objects record for D). Append these activations to the process-activation queue.

Two properties of this approach are worth noting. First, new variable bindings are generated only when new objects are created. This approach is correct because new process activations must include at least one new object — since old objects are never modified, a group of old objects will never spontaneously activate an existing process that had not been activated previously. (Note that a group of old objects could cause a process that was *newly created* by HYPGENE (see Chapter 5) to fire; however, the current HYPGENE implementation does not create new processes.) Similarly, because objects are forked and not deleted, process activations never have to be removed from the process activation list because an object has ceased to exist (since in our model, objects do not cease to exist).

An additional optimization is possible using a slightly different data structure. It may be the case that the interpreter could prove that an existing object O will always prevent process P from firing, because O will always cause a precondition of P to be violated (HYPGENE could prove this by partially evaluating the preconditions of the process [Hsu88]). In this case, the interpreter should never generate an activation of P that includes O . This information could be used in a similar approach that stored live objects within a class on a per-process basis, rather than with every process that acts on the class (the latter is done in the current

live-objects structure). Objects would be removed from the list for a process when GENSIM proved that they could not fire that process.

A Restriction on Process Preconditions

The preceding approach to object management and process execution requires that we impose a restriction on the syntax of process preconditions to guarantee the correctness of the simulator. This restriction has an interesting causal interpretation.

The restriction is that a precondition of a process P may not check for the existence or nonexistence of an object D_1 unless D_1 either is a parameter object of P , or is part of an object that is a parameter object of P . For example, a process P_1 that describes the binding of object A_1 to B_1 may not check whether no objects of class D exist in the simulation:

```
(NOT (EXISTS $X (OBJECT.EXISTS $X 'D)))
```

But it may, however, check whether no objects of class D exist in the simulation as parts of B_1 :

```
(NOT (EXISTS $X (AND (OBJECT.EXISTS $X 'D)
                      (IS.PART $X 'B1))))
```

Without this restriction the correctness of simulations is no longer guaranteed, because the truth of the shorter precondition will depend on *when* the process interpreter evaluates that precondition. If evaluation occurs when no objects of type D exist, then the precondition will be true. But if it occurs after the execution of another process P_2 that creates an object of type D , then the precondition will be false. Thus, the relative execution times of processes P_1 and P_2 — which times are undefined in our simulations — determines the truth of the precondition.

The restriction works because, by stipulating that D_1 must be part of B^1 , it ensures that D_1 must exist at the time P_1 was activated. The second algorithm activates P_1 when, and only

when, all of the parameter objects of P_1 exist. So, if P_2 created B^1 , P_2 must execute before P_1 . Furthermore, given the framework of object forking, once created, B^1 cannot be modified. Thus, there is no possible ambiguity in the evaluation of the preconditions of P_1 .

An important question to ask is: does this restriction have reasonable semantics in the chemistry domain? The answer is yes. In general, chemical interactions are caused when a set of reactants is present in solution, and when each of the reactants is in the correct state. Process preconditions are concerned with evaluating the states of the reactants. In general, the only way one molecule can influence the state of another molecule is by physically attaching to the other molecule and altering its conformation. That is, there is no way for A_1 and B_1 magically to sense the presence or absence of D_1 in solution. To affect the reaction, D_1 must bind to A_1 or B_1 to alter that object's state. Thus, it makes no chemical sense to write the type of precondition we prohibit.

Optimizations

The advantage of this approach to managing simulation objects is this approach meets the described correctness and completeness criteria. The disadvantage is that simulations are slower by roughly a factor of 20 than are simulations in which objects are modified directly. Object forking increases the computational resources required to perform a simulation because some processes within the system generate many complex objects. For example, each movement of RNA polymerase along a DNA strand is accomplished by a different activation of a single process that generates a new copy of the DNA/protein/RNA complex. This forking is costly, both because KEE units are expensive to create, and because the large number of objects generated later yields a large number of process instances. Here we discuss methods for increasing the speed of simulations.

Avoidance of Object Forking In this domain there is a specific case in which objects can be modified directly to avoid the cost of forking the object, without sacrificing the correctness of the simulation. The need to copy-then-modify objects rather than to modify them directly arose from the possibility that multiple processes might act on the original object. Modifying the original object could cause some of its behaviors to go undetected. However, if inspection of the process library reveals that only a single process acts on this object (the object class is named in the `Parameter.Object.Classes` slot of a single process), the preceding consideration would appear to be nullified. Unfortunately, it is not completely nullified, because multiple activations of the process could act on the same object. For example, if we found that the only process acting on the class `RNA-Polymerase` is the `Polymerase.Binds.Promoter` process, the object `RNA-Polymerase.1` still could bind to two different instances of `Trp.Promoter`, such as `Trp.Promoter.1` and `Trp.Promoter.2`. Thus, we cannot avoid forking `Trp.Repressor.1`. We can, however, avoid copy-then-modify when only a single known process can act on an object, and that process acts on *only one* object. For example, the transcription-elongation process does act on a single `Transcription.Elongation.Complex` object whose components are the DNA, protein, and RNA described earlier. This optimization has not been implemented.

One problem this optimization may raise in our domain is that the intermediate objects that we eliminate may aid the HYPGENE hypothesis-formation program in understanding how a theory produced a given prediction. That is, for HYPGENE to understand why the final version of an object has the properties it does, HYPGENE may have to inspect all intermediate versions of the object — which this optimization destroys. Therefore, this optimization may make the learning problem of credit assignment more difficult.

Object Merging A procedure called *object merging* is used to detect when two different processes create different versions of the same object. When this event is detected, only one of the object descriptions is retained. This procedure produces a small economy by eliminating

redundant storage of the merged objects. It produces a much larger savings by preventing processes from being invoked by the redundant object, and by preventing the creation of the additional redundant objects that these duplicate process invocations would produce.

Sharing Object Descriptions with an ATMS It is possible to use an ATMS [DeKleer86] to reduce the computational space complexity of object forking. This approach was inspired by the ATMS property of allowing efficient exploration of alternative decisions through storing rather than recomputing elements of the problem-solving state that the alternatives share. For example, the ATMS has been used previously in qualitative physics to represent envisionments more efficiently [Forbus84,deKleer84]. Our hope was that an ATMS could be used in a similar way to provide more efficient representation of different objects that have a large amount of common structure. This use of the ATMS is novel because we propose using the ATMS to represent more efficiently common aspects of similar objects that coexist within a *single context* of the simulation. Previously, it has been used to represent more efficiently common aspects of similar objects that exist in *alternative* predictions of the state of the physical system (envisionments).

IntelliCorp's KEE contains an ATMS implementation [Intellicorp86,MorrisN86]. In the following paragraphs, I describe this ATMS implementation, sketch how it might be used to solve this problem, discuss why this approach will fail, and then examine additional ATMS functionality that will solve the problem.

KEE's ATMS forms the basis for a facility called *KEEworlds*, which is well integrated with KEE's frame representation system. By default, any modification to a KEE unit (such as modifications to a slot value) affects the *background* (the root world), and all queries access the background by default. New worlds are defined as children of the background and/or existing worlds. Assertions and queries may be explicitly directed to the background or to any existing world. By default, any fact true in the parent of a world *W* is also true in *W*. But *W* may

override *some* of the facts defined in its parents; the values of existing slots in existing units may be modified arbitrarily. However, it is possible to create, delete, or rename both units and slots in the background only.

Figure 3.8 shows how the KEEworlds facility could be used to implement object forking. The background and the worlds W_1 and W_2 represent three consecutive states of the transcription process discussed earlier. In this process, a transcription-elongation complex object contains two other objects: an RNA whose length grows as the process executes repeatedly, and a DNA. Rather than create new versions of the elongation complex object for every step, the KEEworlds facility allows the core descriptions of each object to be inherited by the KEEworlds facility with only the changes to each object recorded explicitly, as shown in Figure 3.8. The substructure of the RNA.1 object changes the length of the RNA increases, and new objects A.1 and B.1 are created.⁴

The limitation of this mechanism arises in the following situation. Imagine the existence of a biological process that specifies that two RNA objects may bind together when they are components of a transcription-elongation complex. If this reaction were to occur between the two versions of the RNA.1 object in worlds W_1 and W_2 , we would create a new world W_3 with parents W_1 and W_2 . W_3 would inherit descriptions of RNA.1 from both W_1 and W_2 . The problem is that the new world W_3 would contain only a single version of RNA.1, whose properties would result from merging the RNA.1 objects from W_1 and W_2 . Chemically, two distinct RNA objects must exist in W_3 , but KEE's approach to world inheritance causes the descriptions of RNA.1 from W_1 and W_2 to be merged, because both objects have the same name. Because object names are both the basis for sharing information between worlds and the source of the merging problem, the KEEworlds implementation is not able to reduce the space required to represent similar objects.

⁴For simplicity, we show the new objects A.1 and B.1 in W_1 and W_2 only, but as noted earlier, new objects must be created in the background.

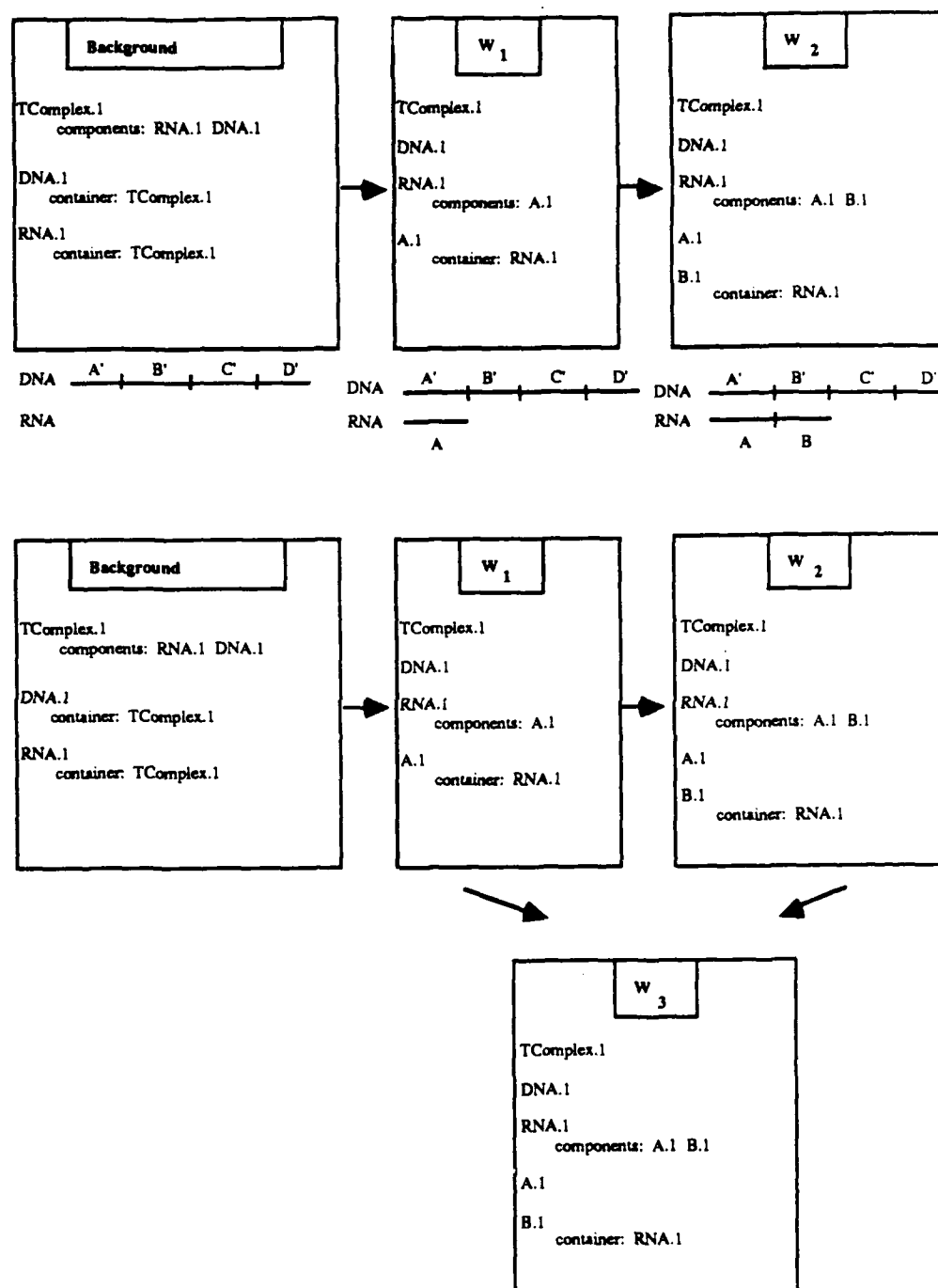


Figure 3.8: Use of an ATMS to share the descriptions of similar objects during a simulation.

This limitation would not exist in an ATMS implementation that had the additional functionality of being able to rename objects from one world to another. That is, if we were able to rename *RNA.1* to *RNA.2* within W_2 , no merging of the two RNA objects would occur in the child world W_3 . GENSIM does not use this technique because we lack such an ATMS. (In KEEworlds, units can be renamed only in the background.)

Other researchers employ an ATMS in their work, but for different purposes: Forbus uses it to represent alternative envisionments more efficiently [Forbus86], and Simmons and Mohammed use it to represent causal dependencies for later analysis by their diagnostic system [SimmonsM87].

3.5.7 GENSIM Results

Figures 3.9 and 3.10 show a GENSIM prediction of all the reactions in the wild-type (normal) tryptophan operon gene-regulation system. GENSIM correctly predicts that an RNA copy of the trp-operon DNA is synthesized through a number of transcription-elongation processes (Figure 7.6 shows the internal structure of one of the transcription-elongation complexes), and that the mRNA thus produced is translated into five different polypeptides. Next, two pairs of the translated polypeptides bind together to form functional enzymes; then, the three enzymes present catalyze the reactions in the trp biosynthetic pathway.

Generation of this prediction required approximately 70 minutes of Dorado (Xerox 1132 LISP machine) CPU time. The prediction contained a total of 1050 objects (including components). The prediction in Figures 3.9 and 3.10 is one of many experiments that have been simulated using GENSIM; Chapter 7 presents other examples.

3.5.8 Conclusions

GENSIM (model 3) provides a framework for representing theories of molecular biology, and for using these theories to predict outcomes of biological experiments. A user describes a theory by

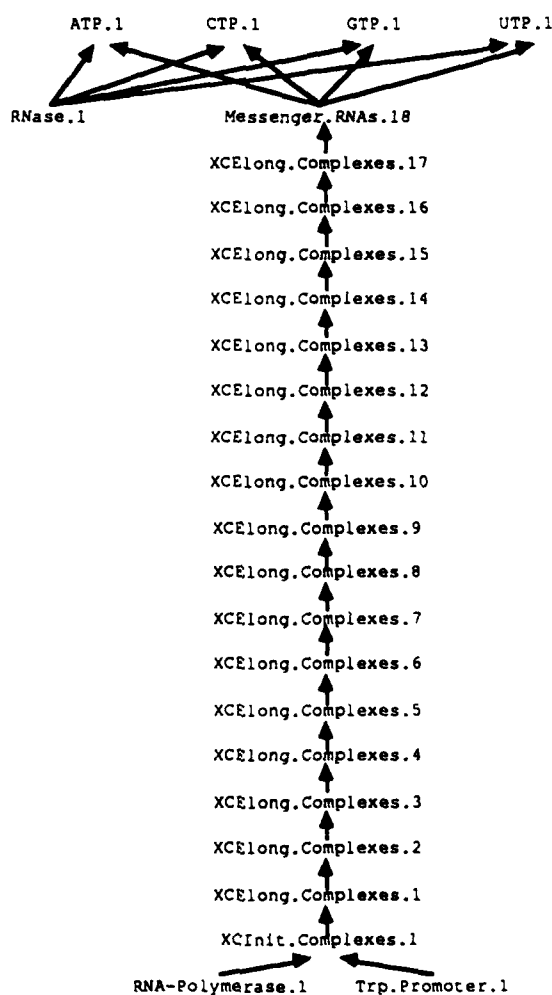


Figure 3.9: Simulation of the *trp* system, part 1. This figure shows a simulation of transcription of DNA and degradation of the resulting mRNA. The nodes of the graph are the names of objects (many objects in this figure contain 10 to 30 component objects. The links in the figure connect the reactants and products of each reaction; for example, *Trp.Operon.1* and *RNA-Polymerase.1* reacted to yield the transcription-initiation complex *XCInit.Complexes.1*.

forking to manage objects during a simulation, and restricts the syntax of process preconditions to eliminate process descriptions that have no valid chemical interpretation. A number of optimizations to the interpreter were discussed, including an efficient process-activation algorithm, a method for merging the descriptions of identical objects that are produced during a simulation, and an analysis of when it is possible to avoid the expensive operation of object forking.

GENSIM is not hindered by the most serious limitation of model 2; the latter uses a fixed network of system state variables that incorporates many assumptions about the physical system being modeled. For example, because it contains the variable `Activated.trpR`, and because this variable is influenced by the variable `Free.trpR`, the system assumes that a functional trp-repressor protein is present in the experiment; but a functional repressor is not present in every experiment. GENSIM reasons explicitly about what objects exist in an experiment, and about what reactions they cause.

Future research should investigate methods for combining the techniques in models 2 and 3 so that quantitative predictions can be made for any experiment. One approach would be to augment process descriptions such that process executions cause new state variables to be linked into a growing state-variable network. This approach would amount to automatic construction of the type of network used by model 2. For example, if a reaction between objects *A* and *B* yielded *C*, a new state variable would be created to represent the concentration of *C*. Functional dependencies would be established among this new variable and those representing the concentrations of *A* and *B*. Once a complete state-variable network was created in this manner (that is, once GENSIM had detected all reactions), methods from model 2 would be used to propagate information through the state-variable network.

Chapter 4

A Conceptual Reconstruction of the Discovery of Attenuation

In 1965, preliminary studies of a set of bacterial genes called the *tryptophan operon* suggested that the expression of these genes was controlled by a gene-regulation mechanism that had been proposed several years earlier to explain the regulation of a set of bacterial genes involved in sugar metabolism. Many scientists expected that this mechanism (called *repression*) would be shown to be responsible for the regulation of most, if not all, bacterial genes. By 1975, however, this view had been shown to be overly optimistic. A long series of experiments demonstrated that, although the tryptophan operon was regulated by repression, its expression was also controlled by a previously unknown mechanism, *attenuation*. These studies led to a detailed understanding of attenuation, which has since been demonstrated to regulate many other genes, both alone and in conjunction with repression [Yanofsky81].

The research performed on the the tryptophan operon from 1965 to 1985 is a rich and complex example of scientific achievement. This chapter describes that biological research from an artificial intelligence perspective. We study the knowledge and reasoning processes employed by the biologists with the objective of understanding the tryptophan research well

enough to build a computer system that can reproduce many of the discoveries made by the biologists.

This study was performed using a combination of techniques from knowledge engineering and information-processing psychology [NewellS72]. We obtained our data by interviewing biologists who performed the research and by reading the scientific publications that the biologists authored. We analyzed these data in two stages. First, we synthesized a *conceptual reconstruction* of the different theories of the tryptophan operon that existed at different points during the research program. That is, we traced the evolution of the theory of attenuation as it was influenced by the accumulation of experimental data. The conceptual reconstruction includes both the path to the current theory of attenuation, and paths to other theories that were eventually rejected. This reconstruction demonstrates that scientific-theory formation, at least for the work studied here, is not a mystical process that will never be flushed from the deep recesses of the human mind, but is instead a concrete, step-by-step process that can be directed by a finite set of strategies and heuristics.

In the second stage, we analyzed the conceptual reconstruction. We examined the differences between successive versions of the theory to learn how new versions of the theory were produced from their predecessors. We studied the experiments performed at different points in time to learn what different strategies of experimentation were employed, and how biologists determined the next type of experiment to perform. Thus, we derived two classes of abstractions: theory-modification operators that are used to generate new theories from existing ones, and modes of scientific exploration that are used to guide the generation of experiments.

In this chapter, we first discuss the methodology used to conduct this study. Next, we examine the organization of the workers in Yanofsky's research laboratory. Then we present the conceptual reconstruction — a detailed chronology of the research that indicates critical experimental and intellectual turning points. The analysis section presents the theory-modification

operators, modes of exploration, and other general patterns detected in the conceptual reconstruction. Chapter 2 presented an overview of general knowledge of molecular biology and of the tryptophan operon, and described the researchers' starting knowledge of the tryptophan operon.

4.1 Study Methodology

The information used in this study was drawn from two principal sources. Personal interviews were conducted with many of the scientists who performed research on the *trp* operon. In addition, we studied over 50 scientific papers that span the course of the tryptophan operon work from 1966 until 1985.

The most in-depth interviews were with Professor Charles Yanofsky, with whom we discussed almost every facet of this research. We also interviewed Dr. Frank Lee (a former graduate student in Professor Yanofsky's laboratory), Dr. Fumio Imamoto (a former coworker of Professor Yanofsky), Dr. Stanley Artz (who worked on the histidine operon), Drs. Robert Landick and Gerard Zurawski (formerly postdoctoral fellows in Professor Yanofsky's laboratory). We have attended three biannual Tryptophan Conferences at which current research related to the *trp* operon is presented.

The papers we studied comprise a large fraction of the original research publications (which appeared in scientific journals) written by Professor Yanofsky's group and by other workers. Some of the papers were review articles that provided a summary of the researchers' view of the operon at a given point in time. We discussed most of the papers with Professor Yanofsky, since we could only partially understand many of them, and since many papers raised interesting questions that were not answered in the original paper. Often papers described a set of experiments without describing why they deemed the hypotheses they were testing likely, or why they did not test other plausible (to us) hypotheses.

The methodology for our study came partly from the studies of human problem-solving behavior performed by Newell and Simon [NewellS72]. In those studies, psychologists observed humans as the latter performed problem-solving tasks such as playing chess or solving cryptarithmic problems. The moves made by the subject were carefully recorded and analyzed, as were verbal records of the subjects "thinking out loud." From these data, the psychologists were able to infer what states of knowledge the subjects attained as they solved problems, and what operators the subjects used to move between these states.

Clearly, the present study deals with different types of data, collected in a posthoc fashion, over a longer period of time, and at longer intervals, than those used in Newell and Simon's studies. For these reasons, and because we are not psychologists, we will not be making psychological claims that are as specific or as detailed as those made by Newell and Simon. We will make claims about what scientists knew at different points in time, and about what hypotheses they were exploring and why. We will also propose mechanisms that are sufficient to account for the generation of new theories, but that may or may not actually have been used by the scientists.

4.2 On Scientific Theories

Chapter 3 discussed scientific theories in some detail. We claim that the representation of scientific theories described there is sufficient but not necessary to model most aspects of the scientific theories discussed in this chapter. That is, the representation in Chapter 3 is largely consistent with what we have observed, but it may not be the only consistent way of representing scientific theories.

The representation contains two basic entities: objects and processes. Objects have various properties that can take on different values; objects are represented as frames. Processes describe object interactions. Process preconditions specify what conditions must be true for

objects to interact; process effects specify what happens when objects interact (e.g., new objects are created). Chapter 3 makes no commitment as to how the credibility of a theory should be represented, so in this respect we do not claim to have a complete model.

Evidence for the claim of representational adequacy is provided in Section 4.5.2, where we show that this framework is able to capture the different modifications made to the trp system during the course of this research. More evidence comes from Chapters 3 and 7, where we show how this approach is used to represent theories of the trp operon.

4.3 Organization of Yanofsky's Laboratory

The researchers who worked in Professor Yanofsky's laboratory composed a skilled team for performing scientific investigations. In this section, we examine the organization of Yanofsky's laboratory to determine how both the attributes of the people in the group and the organization of the group contributed to the success of the researchers.

Yanofsky has been a professor of biology at Stanford University since 1958. His laboratory has investigated many different aspects of tryptophan biosynthesis and gene-structure-protein-structure relationships. Yanofsky's early studies characterized the enzymes in the trp biosynthetic pathway. Since 1969, most of his group's attention has been directed to understanding the regulation of the trp operon of *E. coli* and other organisms.

Between 1969 and 1986, approximately 50 people worked in Yanofsky's group on the *E. coli* trp operon project. Each year, these researchers included visiting professors (1, on average), laboratory technicians (2 or 3), graduate students (3 or 4), postdoctoral fellows (5 or 6), and undergraduates (1 or fewer). On occasion, members of the group traveled to other laboratories to share and acquire new information and techniques (2 or 3 times per year). The size and organization of the Yanofsky laboratory were similar to those of other laboratories working on similar problems, but Yanofsky's laboratory was larger than average, and worked on a more

diverse set of problems at any given time.

Most members of the laboratory attended conferences 1 or 2 times per year; Yanofsky attends 3 or 4 conferences per year. These conferences range from a small biannual conference on the *trp* operon and related systems to large conferences on topics such as transcription, gene regulation, and nucleic acids.

Yanofsky supervised all of the workers in the laboratory by holding weekly or biweekly meetings with each of them, at which times he learned about their findings, and critically evaluated their work and made suggestions for future studies. The researchers met as one large group about 30 times per year to hear progress reports from one member of the laboratory.

In general, these scientists did not work on one set of experiments as a team. Among the reasons for this isolated style of research was that different people work at different speeds, and thus did not want to be hindered by a slower colleague. Also, such cooperation could give rise to conflicts over who was responsible for ideas or results and who would take credit for them. Yanofsky's belief is that in an ideal world, his laboratory would function more efficiently if people could cooperate without these concerns [Yanofsky87].

One obvious attribute of the laboratory was its highly distributed organization — several technicians, students, and postdoctoral fellows worked in parallel on different but related problems. One consequence of this organization is that individual workers are often not aware of one another's recent accomplishments and goals, making it less likely that one worker will become distracted by the experimental problems and anomalies encountered by another worker. During the work on the *trp* system, there were periods when a number of unrelated anomalies existed due to a number of errors in the existing theory. At such times, it seemed most profitable for a problem solver to focus her attention on single problems or on a related subset of all outstanding problems at a time. Yanofsky, however, coordinated the actions of all the people in the laboratory, attempting to comprehend and synthesize all their results and anomalies. This broad perspective allowed Yanofsky to understand a single underlying mechanism that

manifested itself in the work of several people. In the mid-1970s, for example, several members of Yanofsky's laboratory each found different pieces of the puzzle of attenuation; Yanofsky assembled the mechanism of attenuation in its entirety because only he saw several of its pieces simultaneously.

4.4 Annotated Chronology of the Research

In this section we provide a detailed history of the research that led to the discovery and understanding of attenuation. This chronology is partitioned into five states of knowledge through which the researchers passed in their study of this biological system. We define a *state of knowledge* as a reasonably stable, coherent theory of the regulation of the trp operon that existed for roughly one year or more, and that was distinct from states that preceded and followed it. Such states can be viewed as conceptual ports of call, or way stations, on the long quest to understand the trp operon. Of course, within each of these states new research was being performed continually, so the notion of a stable state is a simplification. Figure 4.1 provides an outline of these states of knowledge.

Associated with each of the five states is a description of the research that produced the next state of knowledge. As described in Section 4.1, we prepared these descriptions by interviewing scientists and by studying the scientific literature they authored. This description of the research is organized around that literature; we describe the experiments and results reported in 41 key papers from scientific journals (at least 90 percent of the papers describing research on regulation of the trp operon in the period from 1965 to 1985). We use the term *unit of research* to refer to each of the sets of experiments described in this section.

This conceptual reconstruction is long and detailed. Therefore, we have provided several figures that summarize it, to which we shall refer throughout the chapter. At this stage the intent is only to introduce these figures and provide an overall perspective for the details that

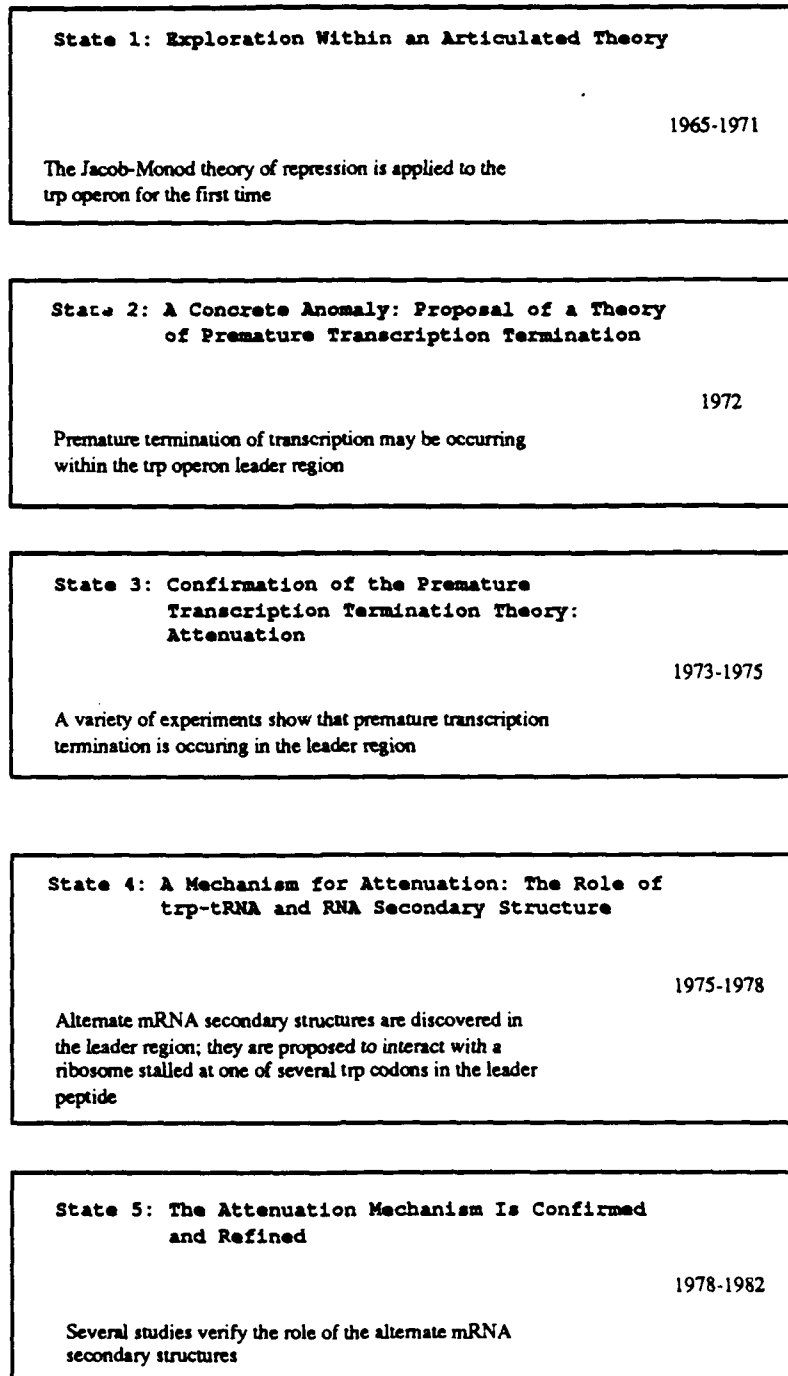


Figure 4.1: The major states of knowledge possessed by the trp system researchers.

follow. We will discuss them in more detail later, so the reader should not attempt to fully understand them now.

- Figure 4.2 summarizes how units of research have been partitioned into the five states of knowledge; it can be used as a reference to locate a given paper.
- Figure 4.3 groups papers by subject matter, showing related papers in a series of studies on the same subject, and how studies in one area gave birth to studies in another. For example, all the sequencing studies of the *trp* operon are connected by arrows. The vertical axis of this figure represents time, so it also shows when different studies were performed relative to one another.
- Figure 4.4 outlines the development of the theory of the *trp* operon, and is analogous to the evolutionary tree for a family of organisms. Evolutionary trees indicate how the descendants of an organism diverge to form new species, and how some species die out, forming a dead-end path in the tree. Each node in Figure 4.4 corresponds to a distinct version of the theory. Edges of the graph are labelled by the names of the research unit(s) that caused the theory to change from one form to another.
- Figures 4.5 and 4.6 are similar to Figure 4.3, but they list the alternative hypotheses that the researchers considered.

We begin with a synopsis of the entire course of the *trp* operon research. Then we describe each of the five states of knowledge in considerable detail. Each state description summarizes what was known about the *trp* operon at the start of that state, then describes the research carried on within that state and how it affected the theory of the *trp* operon. Readers who find the detailed state descriptions too intense are advised to read only the synopsis, the starting state descriptions, and a handful of the research-unit descriptions, such as those for [ForchhammerJY72] (state 1), [JacksonY73] (state 2), [SquiresLY75] (state 3), and [OxenderZY79] (state 4).

<u>State 1</u>	[Hiraga69]	[Imamoto70]	[YanofskyH72]	[ItoHY69]
	[ForchhammerJY72]	[CohenYY73]	[JacksonY72]	[BakerY72]
	[RoseY72]	[LewisA71]		
<u>State 2</u>	[JacksonY73]	[Kasai74]		
<u>State 3</u>	[ArtzB75]	[PlattY75]	[PlattSY76]	[SquiresLY75]
	[LeeSSY76]	[BertrandY76]	[BertrandSSY76]	[SquiresLBSBY76]
	[KomY76]			
<u>State 4</u>	[MorseM76]	[YanofskyS77]	[BertrandKLY77]	[LeeY77]
	[MiozzariY78a]	[ZurawskiBKY78]	[BennettY78]	[BrownBLSY78]
	[BennettSBSY78]			
<u>State 5</u>	[StroynowskiY82]	[StroynowskiKY83]	[DasCY82]	[ZurawskiY80]
	[Platt81]	[NicholsVY81]	[Yanofsky83]	[KelleyY82]

Figure 4.2: Each paper discussed in this chapter is assigned to one of the major states of knowledge.

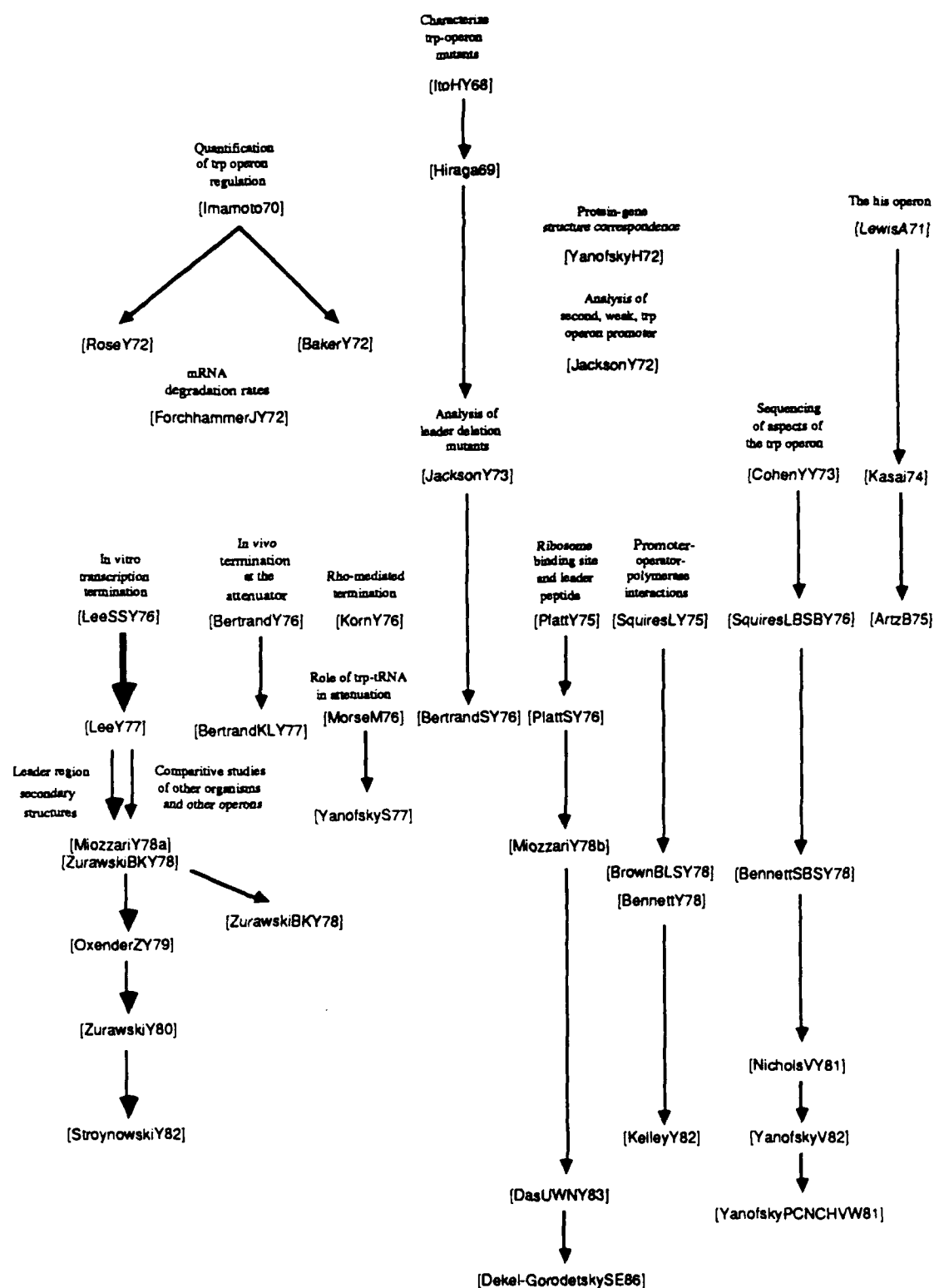


Figure 4.3: The *trp* literature is organized by time along the vertical axis; arrows indicate papers that are related by subject matter. The small headings indicate a topic, and all papers connected by arrows of a given type are about the same topic. A few papers, such as [LeeY77], are grouped within more than one topic.

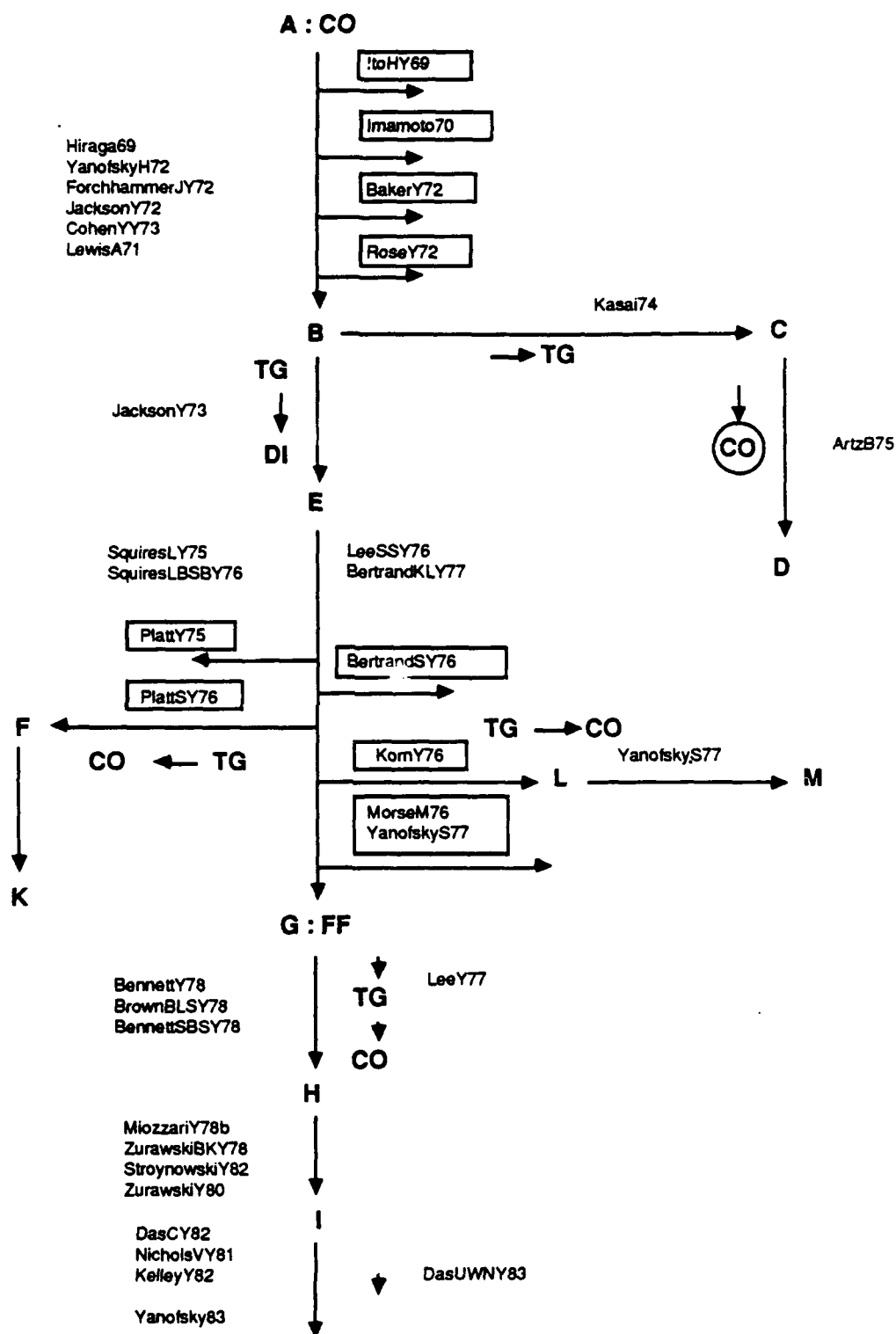


Figure 4.4: The evolution of the theory of the regulation of the tryptophan operon. Each node in corresponds to a distinct version of the theory. Edges of the graph are labelled by the names of the research unit(s) that caused the theory to change from one form to another.

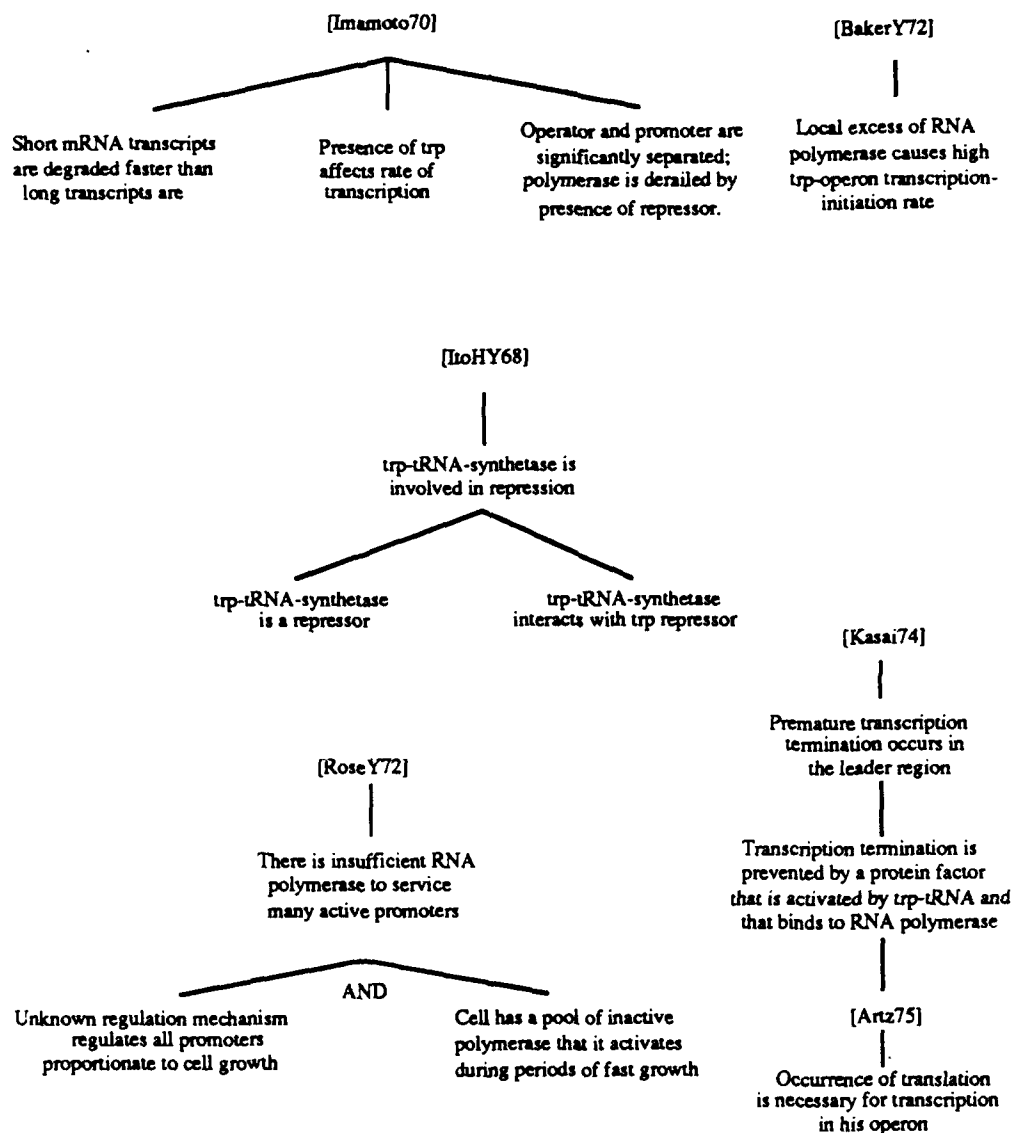


Figure 4.5: The evolution of the theory of the regulation of the trp operon. The figure summarizes alternative theories of trp operon regulation that were considered by the trp researchers.

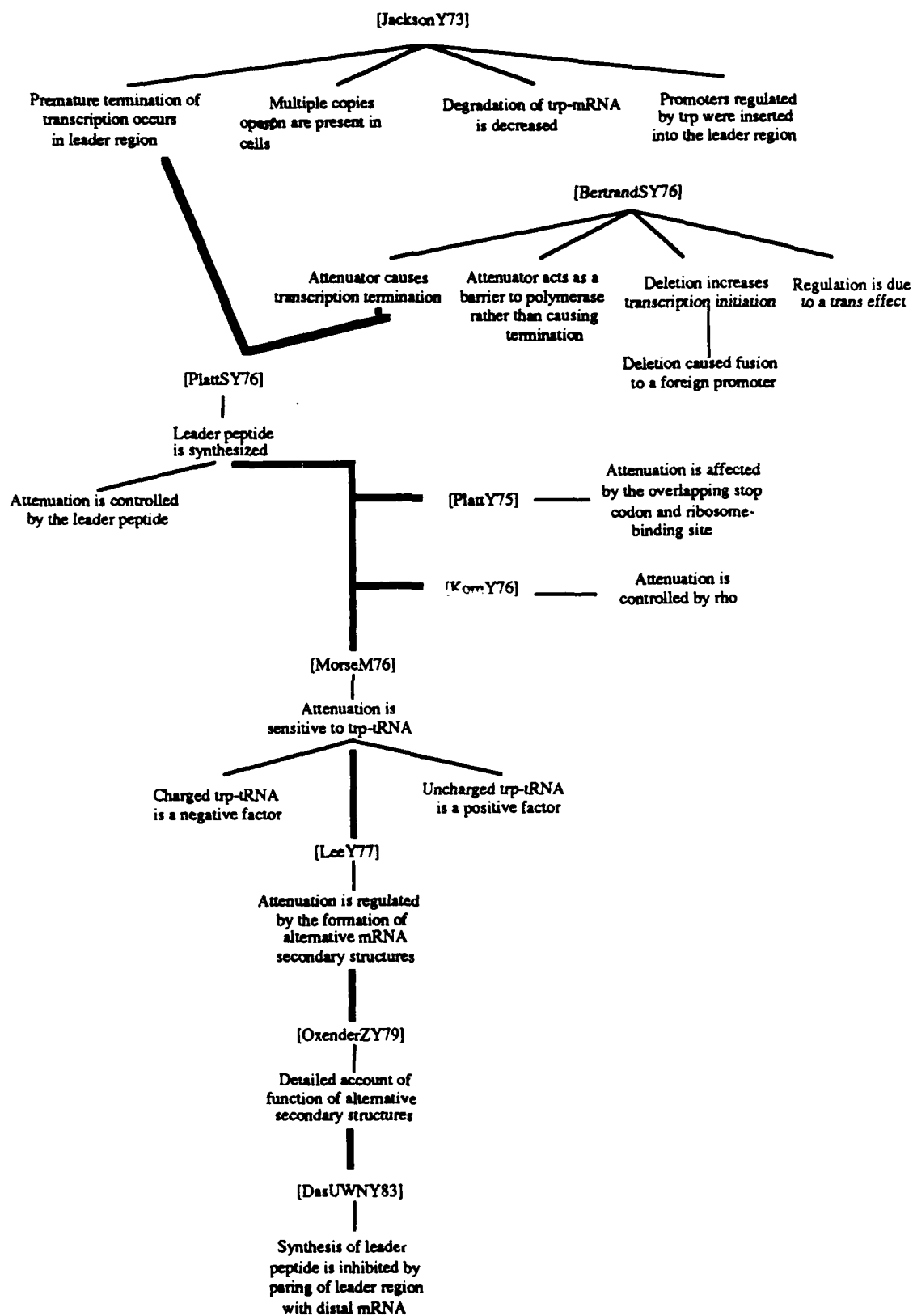


Figure 4.6: The evolution of the theory of the regulation of the trp operon. Alternative theories of trp operon regulation that were considered by the trp researchers.

Referring to the dates of particular events is complicated by the significant time lag between the occurrence of an event in the laboratory and the publication of a description of the event in a biological journal (the lag ranges from months to years). For simplicity, we shall refer to “publication time” by default, and will note explicitly when a date is in “laboratory time.”

4.4.1 Research on the Trp Operon: Synopsis

Before proceeding with the history, we shall note some of the general properties of the process diagrammed in Figures 4.3 and 4.5 and 4.6. One interesting property of the research that is evident from Figure 4.3 is that, although there are a significant number of branches from the main path, all these deadends were abandoned or demonstrated to be implausible after at most 2 years. The B-C-D branch is the route taken by researchers who explored a different *E. coli* operon — the *his* operon — that codes for enzymes that synthesize the amino acid *histidine*. A number of other *potential* branches are shown with the name of the corresponding research unit in a box. Later sections describe why these were potential branch points, and why they were not taken. Of the 31 theories listed, 17 were incorrect, meaning that none of the novel attributes of these theories exist in the current theory of the *trp* operon. Whether this fraction represents good, bad, or average performance is difficult to assess; we are aware of no other data on this question.

We now proceed with the history itself, using Figure 4.3 as a guide. State A is the initial theory of the *trp* operon — the theory of repression developed in the *lac* operon developed by Jacob and Monod [JacobM61]. Preliminary experiments on the *trp* operon had indicated that repression theory would be applicable to the *trp* operon as well as to the *lac* operon. This hypothesis was verified by a number of researchers who characterized different parameters of the *trp* system in detail; for example, [Hiraga69] observed expected classes of *trp*-operon mutants. However, minor anomalies were detected by [ItoHY69,Imamoto70,BakerY72,RoseY72]. They

made detailed measurements of the expression of the operon that contradicted the Jacob-Monod theory to a degree. For example, [BakerY72] observed unexpectedly high transcription-initiation rates when cells were transferred from media containing trp to trp-free media. Each of these anomalies could have been pursued with additional experiments to determine their causes. Such experiments could have given rise to modifications in the theory of the trp operon. The horizontal arrows out of state A indicate the potential for modifications to the theory because of the anomalies detected in these experiments.

When state B was reached, the operon had been characterized in some detail, and minor inconsistencies between theory and observation had been detected. Major inconsistencies were revealed by the studies of [JacksonY73,Kasai74]. These researchers deleted segments of DNA from the leader regions of the trp operon and the similar his operon and observed a large increase in expression of the respective operons. This increase was not predicted by the traditional repression theory. At this point, the paths diverged. Kasai postulated the existence of a protein that attached to RNA polymerase at transcription initiation and prevented transcription termination at the leader region site he deleted. This new protein was in theory activated by charged tRNA^{his}. Jackson performed additional experiments to distinguish among several theories that would explain the increased operon expression. Her experiments indicated that premature termination of transcription was occurring in the leader region at the site she had deleted, but she did not propose a specific mechanism of termination.

Artz followed up on Kasai's experiments and obtained results that confirmed Kasai's antitermination-factor theory to a degree [ArtzB75]. State D represents Artz' experiments, after which the work in the his operon was not pursued until the mechanism of attenuation was understood in the trp operon.

From state E — after Jackson's experiments — Yanofsky's laboratory performed two sets of studies. They first repeated and elaborated on Jackson's results to provide more evidence that trp-sensitive premature termination of transcription was indeed occurring in the leader

region [BertrandKLY77,BertrandSY76,LeeSSY76]. The members of the second set of studies were largely unrelated and sought to further characterize general properties of the trp operon, such as its DNA sequence [SquiresLBSBY76], and the locations of its ribosome-binding sites [PlattY75,PlattSY76]. This latter work uncovered several anomalies. Platt found a ribosome-binding site in the leader region of the operon, which indicated that the leader region produces a small protein, although no such protein was known. This observation led the researchers to propose that the leader peptide regulates attenuation, and spurred Platt to search for this protein in vivo — but he did not find it (state F). [MiozzariY78b] later demonstrated that the leader-region ribosome-binding site is active (state K).

Korn obtained results suggesting that the rho protein (responsible for polarity) was involved in attenuation. He pursued these studies further and found that rho was in fact *not* involved (state L); his earlier studies had been misleading because of an undetected mutation, which was later characterized in [YanofskyS77] (state M).

A third anomaly became evident when DNA sequence analysis revealed the presence of two consecutive trp codons in the leader region of the operon. They were downstream of the leader-region ribosome-binding site and thus would have been present in the protein for which Platt was searching. Their presence was significant because tryptophan is generally a rare amino acid in proteins (only 1 percent of the amino acid residues in proteins are tryptophan residues). The existence of two tryptophans in a row is improbable, and the fact that they were found in the trp operon was even more interesting.

Two other studies confirmed an earlier suspicion that attenuation was sensitive not to the concentration of trp itself, but rather to the concentration of charged tRNA^{trp} [MorseM76,YanofskyS77].

At the start of state G, the trp system had been characterized in considerable detail, and the theory of premature transcription termination in the leader region was fairly well accepted. No molecular mechanism for the termination of transcription was known, however,

and a few anomalous facets of the trp system remained. Lee resolved these issues by performing enzyme-digestion experiments to search for secondary-structure folding patterns in the leader-region mRNA. He found evidence for the existence of secondary structures, and computer analysis of the DNA sequence of the region indicated that two alternative secondary-structure configurations could form there.

Lee's result acted as a catalyst that crystallized a theory of the mechanism of attenuation. The two alternative secondary structures are analogous to the poles of a switch: in one configuration, they signal RNA polymerase to terminate transcription; in the other configuration they allow it to continue transcription. The state of the switch is determined by the presence or absence of a ribosome on a specific area of the leader region mRNA. Ribosomes attach to a growing mRNA and synthesize protein as the mRNA is synthesized. The rate of ribosome movement is determined by the concentration of charged tRNA^{trp}: When there is little trp in the cell, there is little charged tRNA^{trp} (bound to trp); thus, the ribosome *stalls* at the trp codons in the leader region, waiting for some of the scarce charged tRNA^{trp} to attach. When the ribosome stalls it is not close to the mRNA secondary structures, thus allowing the *antiterminator* secondary structure to form. When the ribosome does not stall, its presence blocks formation of the antiterminator structure, thus allowing the *terminator* secondary structure to form. This process is diagrammed in Figure 4.7.

This theory was confirmed and elaborated on in states H through J. Several studies investigated the properties of the secondary structures in intricate detail. Experimenters showed that various structural features of the *E. coli* trp operon could also be found in the trp operons of other bacteria, and in *E. coli* operons for other amino acids. In particular, Zurawski found seven consecutive phenylalanine (*phe*) codons in the leader region of the phe operon [ZurawskiBKY78]. In addition, the rate of transcription termination was shown to correlate with the stability of the secondary structures [StroynowskiY82,ZurawskiY80].

Yanofsky's laboratory continued to characterize the trp system in even greater detail; for

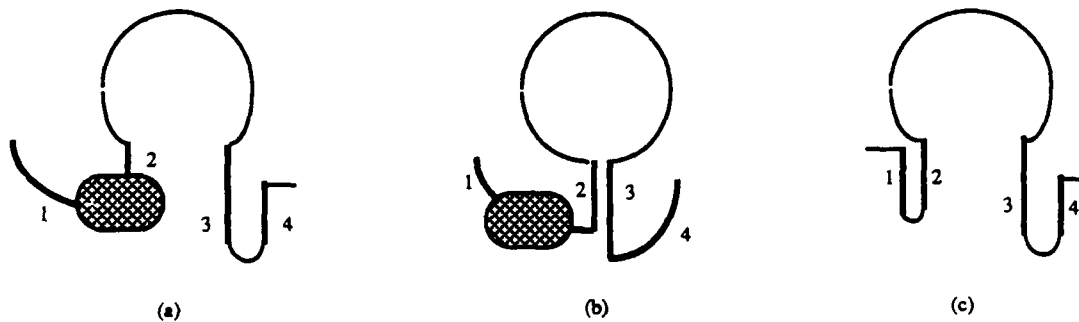


Figure 4.7: Alternative secondary structures in the *trp* operon leader region. Base pairing can take place between four different regions within the leader region; region 2 can pair with regions 1 or 3, and region 3 can pair with regions 2 or 4. These secondary structures are mutually exclusive. (a) Under conditions of excess *trp*, the ribosome does not stall at the *trp* codons, so the terminator structure 3:4 can form — termination occurs. (b) Under *trp* starvation, the ribosome stalls, allowing structure 2:3 to form and preventing the terminator structure from forming — no termination occurs. (c) When no translation occurs, structure 1:2 forms, which allows the terminator to form — termination occurs. [OxenderZY79]

example, the positions of the *trp* operator and promoter were established more precisely [BennettY78,BrownBLSY78], and the *trpE* gene was sequenced [NicholsVY81]. [DasUWNY83] showed that the leader peptide could be synthesized in vitro, and [Dekel-GorodetskySE86] detected its synthesis in vivo. Yanofsky's group also determined that the relative positions of the ribosome and mRNA are synchronized by an elaborate mechanism, the description of which is beyond the scope of this paper [LandickCY85].

We now present the detailed conceptual reconstruction of the *trp* research, which is divided into five main states of knowledge.

4.4.2 State 1: Exploration Within an Articulated Theory

Description of Beginning of State

This first state existed when this research began, and was described in Chapter 2. The biosynthetic pathway for trp had been well understood for several years, but knowledge of the molecular genetics of the system was sketchy. The polypeptide products of the operon were known, as was the layout of their genes within the operon, and the fact that they were regulated as one unit. It was known with some certainty that they were regulated by Jacob-Monod repression, although the details of this regulation had not been probed. Only one other operon had been examined in detail: the lac operon, studied by Jacob and Monod.

State Transformations

[Hiraga69] — Characterization of Constitutive Mutants

[Hiraga69] describes a number of experiments in which different mutants of the trp system were characterized. The researchers selected for constitutive trp operon mutants, then mapped the locations of these mutants. Some of the mutations were at the beginning of the operon, and were assumed to be operator mutations. Others were at a different genetic locus, and were assumed to be repressor mutations. The authors also attempted to estimate the size of the operator region and its distance from the first gene in the operon, *trpE*. In 1969, it was practically impossible to map DNA precisely, so even this relatively coarse information about locations of genetic regions within the operon was of significant value.

[Imamoto70] — Studies of the Time Course of the Onset of Repression

[Imamoto70] describes experiments that explored the time course of the onset of repression. Imamoto added tryptophan to derepressed cells and then measured their *trpE*-mRNA levels after 1 minute, and at 10 second intervals thereafter. This technique allowed him to observe,

with a fine time resolution, the decrease in *trpE*-mRNA production due to repression. *trpE*-mRNA is also degraded over time due to the cell's normal mRNA degradation mechanisms. Based on the (approximately known) length of the leader region and *trpE* region of the operon, Imamoto could calculate when the *trpE*-mRNA production should stop completely.

Imamoto observed less *trpE*-mRNA than expected — *trpE*-mRNA production dropped off sooner than the repression theory predicted, given known rates of transcription elongation. Thus, it appeared that the transcripts that were being synthesized at the time *trp* was added to the cells were disappearing. (The transcripts disappeared because the added tryptophan activated the attenuator and prematurely terminated some transcription complexes, making this experiment an early observation of attenuation.) Imamoto formulated a number of hypotheses to explain why transcription stopped so soon. One hypothesis was that short transcripts were degraded more rapidly than longer transcripts were (this hypothesis was tested and discarded). Another hypothesis was that the addition of tryptophan accelerated the rate at which polymerase transcribed mRNA; this was also discarded. The last hypothesis was never tested: Imamoto postulated that the *trp* operator was significantly downstream of the *trp* promoter, and that when the repressor bound to the operator, this binding “derailed” transcription complexes that approached the repressor from the upstream promoter.

In fact, the operator is not downstream of the promoter. Had experiments been performed to detect the prematurely terminated transcripts, however, they might have been successful — the attenuator produces such transcripts.

At the time, Imamoto considered another hypothesis as well; namely that termination of translation was occurring, and that this termination in turn was causing premature termination of transcription by the mechanism of polarity [Imamoto84].

[ItoHY69] — Studies of *trp*-tRNA-synthetase Mutants

The work of Ito et al. characterized an *E. coli* mutant designated *trpS5*. The mutation is located outside the *trp* operon. This mutant requires *trp* to grow, suggesting that the *trp*

Strain	Growth*		Doubling time (min)		Tryptophan synthetase**	
	-Trp	+Trp	-Trp	+Trp	Derepressed	Repressed
Wild (Y-mel)	+	+			100	3
trpS5	-	-				
trpO1	+	+			165	41
trpO1 trpS5	+/-	+	89	54	305	98
trpR970	+	+			168	145
trpR970 trpS5	+/-	+	142	61	58	47
trpE5927-1	+	+			109	9
trpE5927-1 trpS5	+/-	+	139	65	N.T.	N.T.

* +, -, +/- represents growth, no growth, or slow growth, respectively, on minimal agar with or without L-tryptophan.

** Values represent percentages of the derepressed wild-type activity using cell suspensions for enzyme assay.

Table 4.1: Experimental data from [ItoHY69] that shows the rate of growth of different *E. coli* mutants in different media. *trpS5* is a *trp*-tRNA-synthetase mutant; *trpO1* is a *trp*-operator mutant. Tryptophan synthetase is an enzyme produced by the *trp* operon; the amount of this enzyme present indicates the rate at which the *trp* operon is expressed.

operon is turned off. An assay found no *trp*-tRNA-synthetase activity, and extracts from these cells did not inhibit the synthetase activity of normal cells (indicating these mutants were not producing an inhibitor of *trp*-tRNA-synthetase). Thus, the researchers concluded that *trpS5* is a *trp*-tRNA-synthetase mutant.

One experiment showed that *trpS5* mutants had constant, elevated levels of *trp*-operon expression, relative to wild type, when *trp* is not present in the medium.

Another interesting set of experiments is shown in Table 4.1. Comparison of the row *trpO1 trpS5* (a mutant with both a bad operator and the *trpS5* mutation) with the *trpO1* row shows that the strain with the malfunctioning synthetase has elevated levels of operon expression in both the repressed and derepressed states (by a factor of 2), thus suggesting that there is a regulation mechanism present that is separate from repression and is somehow related to *trp*-tRNA-synthetase.

In addition, the *trpO1* row is of interest of itself, since even in a *trpO*- mutant, partial regulation of expression (a factor of 4) is observed under different *trp* concentrations. We could explain this effect by postulating that the *trpO* mutation only partially damaged the operator.

However, if it was known that the operator was completely destroyed, then another regulation mechanism had to exist. More evidence for an additional regulation mechanism could be obtained by examining *trpR*– strains. These experiments were also performed, yet their results are in disagreement with what we would expect. The *trpR970* row shows expression that is independent of *trp* concentration, whereas we would expect that, with repression knocked out, some regulation due to attenuation would be observed, which would be an important effect. The *trpR970/trpS5* row shows markedly decreased expression relative to wild type, whereas we would expect to see extremely elevated expression.

Note that the (important) data for the *trpS5* mutant have been omitted from this table. Our theory as to why it was omitted is that, in the repressed state this mutant's level of expression was similar to that of wild type, but the researchers could not explain the equal levels of expression. Now, Yanofsky would explain this level of expression by postulating that, even though the mutant synthetase is only partially functional, with excess *trp* present it is sufficiently functional to behave like wild-type synthetase; its deficiency appears only when the cell is starved of *trp* [Yanofsky87].

Although these experiments provide a possible path to the discovery of attenuation, it is unclear exactly how a scientist would proceed from these results. The reason that it is not clear what future experiments should be performed is that the biologists had no knowledge that provided any clue as to how tRNA^{trp} might affect the regulation of the *trp* operon [Yanofsky87,Hiraga84].

[ForchhammerJY72] — Quantification of *trp*-mRNA Degradation Rate

Forchhammer was interested in general aspects of mRNA degradation. This paper describes studies of the half-lives of mRNA regions at different locations along the *trp* operon. The scientists found that operator-distal regions had longer half-lives, but that their lifetimes were determined not by their primary sequence, but by their position within the mRNA transcript of the entire operon: Promoter-proximal regions had shorter half-lives. These observations led

to a model for mRNA degradation; namely that it proceeds in the 5' to 3' direction. mRNA degradation had been studied previously; these results agreed with some of the previous work and disagreed with some of it. This study provided detailed information of fine resolution about an element of the *trp* system that had not been well understood previously.

[CohenYY73] — Sequence of 5' End of *trp*-mRNA

[CohenYY73] discusses experiments whose purpose was to sequence the 5' end of the *trp* operon mRNA (from the promoter into the *trpE* gene). This region was of interest because it included the promoter, the operator, and the ribosome-binding site within the *trpE* gene. During this period it was not possible to sequence DNA. Yet, RNA sequencing was not trivial either, and much of the experimental work was devoted to overcoming problems with this technique. The researchers obtained the sequence of an operator-proximal segment 110 bases in length, but were unsure of its 5'/3' orientation and its position within the operon. A more accurate version of the leader-region sequence was published in [SquiresLBSBY76], and the full sequence of the operon was obtained in 1980 [YanofskyPCNCHVW81].

[BakerY72] — Quantification of Operon Transcription and Translation Rates

[BakerY72] describes detailed measurements of the transcriptional and translational yields of the *trp* operon, allowing determinations of the rate of transcription initiation and the protein yield of each *trp*-mRNA before degradation. The investigators obtained the following results:

1. There were 2.6 initiations per operon per minute in the *trpR*- strain
2. There were 5.1 initiations per operon per minute in the *trpR*+ strain immediately after derepression (cells were placed in a *trp*-free medium)
3. There were 1.1 initiations per operon per minute in derepressed *trpR*+ strain steady-state (minimal amounts of *trp* were present, synthesized by the cell)

Result 2 was anomalous, because result 1 should have represented a ceiling on the transcription initiation rate if repression was the sole transcription regulatory mechanism; *trpR*-

strains have no functional repressor, so the operon should have been turned on all the time. Yet a *trpR*⁺ strain with an inactivated repressor (due to no tryptophan in the environment) showed a higher initiation rate.

In fact, we now know that this effect is due to relief of attenuation by *trp* starvation. The researchers justified result 2 by postulating that a “local excess of RNA polymerase” existed at the *trp* operon, although this hypothesis was not taken seriously at the time, and could not be tested experimentally [Yanofsky87].

[RoseY72] — *trp*-mRNA Synthesis is Dependent on Richness of Growth Medium

[RoseY72] presents measurements of *trp*-mRNA synthesis made when *trpR*[−] strains were grown in different media (none of which contained *trp*). As the media became richer (up to a point), more and more *trp*-mRNA was produced, in proportion to the cell growth rate. When richness increased past a limit, however, *trp*-mRNA production decreased as the cell growth rate increased. The increased *trp*-mRNA was determined to be due to an elevated rate of transcription initiation, not to changes in rates of elongation or degradation.

The researchers explained the decreasing *trp*-mRNA by postulating that, at high growth rates, the large number of active promoters in the cell were diverting RNA polymerase away from the *trp* operon. The increasing effect could not be explained, however, since the only known regulation of the operon was inactivated. Rose and Yanofsky postulated an unknown regulation mechanism by which all promoters in the cell were coordinated with cell growth rate.

[LewisA72] — His Operon Expression is Sensitive to tRNA^{his}

During this period, researchers in another laboratory were exploring the regulation of the *his* operon, which codes for enzymes that form a biosynthetic pathway for the amino acid histidine. [LewisA72] was one of a series of papers on this subject. Previous experiments had suggested that the *his* operon was also regulated by repression, and that *his*-tRNA-synthetase (the enzyme responsible for charging tRNA^{his}) was the repressor protein that regulated this

operon.

The experiments described in this paper show that the expression of the *his* operon is sensitive to the concentration of charged tRNA^{his}. Thus, the authors proposed that tRNA^{his} and his-tRNA-synthetase bind together to form an activated repressor for the *his* operon. However, wild-type tRNA^{his} and a mutant tRNA^{his} that caused constitutive synthesis of the *his* operon showed the same degree of binding to his-tRNA-synthetase. This observation of equal binding was fairly strong evidence that his-tRNA-synthetase is not a repressor of the *his* operon, since the mutant tRNA^{his} does not repress the operon, but does bind to his-tRNA-synthetase (an event that would usually accompany repression). The authors noted, however, that binding between the two might take place, yet not produce a required change in the conformation of the repressor protein.

4.4.3 State 2: A Concrete Anomaly and Preliminary Discrimination of Its Causes

Description of Beginning of State

At the start of state 2, in 1972, a number of exploratory experiments had been performed on the *trp* operon that quantified several aspects of the system, such as rates of mRNA degradation and transcription initiation, and the genetic structure of the operon. These experiments also uncovered a number of anomalous findings:

- Imamoto observed that transcription stopped sooner than expected during the onset of repression [Imamoto70].
- Baker observed a very high transcription-initiation rate in severely *trp*-starved cells [BakerY72].
- Rose observed that *trp*-mRNA synthesis in *trpR*- strains increased and then decreased as the cell growth rate increased steadily [RoseY72].

- Ito and his colleagues observed that trp-tRNA-synthetase played a role in regulation of the trp operon [ItoHY69]. Hiraga indicated that he believed there may have been a binding site for tRNA^{trp} near the operator, which related to a mode of regulation in addition to repression [Hiraga84]. Ames' group had made similar observations regarding the his operon [LewisA72].

State Transformations

[JacksonY73] — Leader-Region Deletions Elevate Derepressed Transcription Levels

[JacksonY73] provided the first detailed, solid evidence that the workings of the trp system were not fully understood. These experiments examined the behavior of six mutants with large deletions that removed the *trpE* gene and leader region of the operon; the *trpB* and *trpA* genes were fully intact. Two of these strains had higher maximal levels of expression of the intact genes than wild-type strains, and two had lower maximal levels of expression. (The biologists observed maximal expression levels by including *trpR*⁻ mutations in these strains to disable the cell's repression machinery). The remaining two mutants had essentially unchanged levels of expression.

Jackson and Yanofsky proceeded to characterize the strains in detail. When all strains were grown in excess trp, significant differences were observed between rates of expression of strains containing wild-type *trpR*, and those containing a *trpR*⁻ strain, in addition to the leader region deletion. Thus, the operator region was still intact.

The researchers explained the mutants with decreased maximal levels of expression by postulating that the deletions produced a shift in the translation reading frame and introduced a translation stop codon. This codon would cause premature translation termination, which would cause premature transcription termination by the mechanism of polarity, thus decreasing gene expression.

The researchers next attempted to account for the two mutants with the elevated maximal levels of expression. Since the initial experiments had measured production of the trp biosynthetic enzymes, additional experiments were performed to measure the synthesis of trp-mRNA. Its synthesis was found to be elevated, suggesting that the deletions were altering the process of transcription, not that of translation. Another series of experiments explored possible reasons for the elevated levels of trp-mRNA. One explanation was that the rate of cellular degradation of mRNA is altered in these mutants. mRNA degradation rates were compared and were found to be essentially the same, or perhaps slightly lower in the mutants than in the wild type. The elevated mRNA levels would also result if the mutants carried multiple copies of the trp operon. This hypothesis was checked and rejected. Another hypothesis was that the mutants had a second promoter downstream of the wild-type promoter, created by these deletions. This possibility was not ruled out completely, but the researchers showed that the deletions mapped to the right of the trp operator.

The authors concluded that a site exists between the operator and *trpE* whose normal function is to decrease maximal expression of the operon, and that the deletions had removed this site. At this time, it was not known that this site was sensitive to trp concentration. The biologists proposed that the site affected continuation of transcription, and that it might be regulated by a metabolic compound.

[Kasai74] — A New Gene-Regulation Mechanism Proposed for the his Operon

This paper describes experiments with several mutants in the his operon. The mutant *hisO1242* is similar to some of Jackson's mutants [JacksonY73] in that the former shows elevated operon synthesis in the presence of histidine. Because the his operator is downstream of the promoter, Kasai concluded that this mutation deleted a barrier to transcription elongation, which he called the *attenuator*.

Three other classes of mutants in the leader region (called I, IIA, and IIB) were studied in conjunction with a tRNA^{his} mutant called *hisT1504*, and with *hisO1242*. The tRNA^{his}

mutant also elevated expression of the operon in the presence of histidine. These different combinations of mutants showed different patterns of expression; for example, IIB and *hisT1504* showed wild-type levels of synthesis, whereas IIB and *hisO1242* showed elevated levels of synthesis. Thus, IIB appeared to block the effects of *hisT1504*.

Kasai postulated the following mechanism to explain these results: tRNA^{his} recognizes a region of the *his* operon DNA and regulates the binding of a protein factor at the promoter. This factor accompanies RNA polymerase during transcription and prevents the attenuator from terminating transcription. Later experiments showed that this model was incorrect, and that the *his* operon was regulated by the same attenuation mechanism as was the *trp* operon.

[ArtzB75] — Further *his* Operon Studies

[ArtzB75] confirmed Kasai's work and elaborated it to produce three principal results. First, this study again demonstrated that a mutant tRNA^{his} can cause a wild-type *his* operon to become constitutive.

Second, *in vitro* studies showed that normal transcription of the wild-type *his* operon will occur *in vitro* only if an *in vitro* translation system is present as well. For the *hisO1242* mutant, however, the translation system is not required for transcription to occur.

Third, a titration experiment showed that *in vitro*, as the concentration of wild-type DNA is increased, operon expression reaches a limit, whereas for the *hisO1242* strain, expression increases linearly. This third result suggested that a positive factor required for operon expression was being exhausted at the high DNA concentrations, and thus supported Kasai's model of attenuation. The paper concluded that a positive factor was interacting with the attenuator in some way to prevent premature termination of transcription. (Hindsight suggests that this "positive factor" was the ribosome [Artz86, Yanofsky87].)

4.4.4 State 3: Confirmation of the Theory of Premature Transcription Termination

Description of Beginning of State

Jackson's experiments established a concrete discrepancy between the predictions that the *trp* system was regulated by repression, and the behavior of the system [JacksonY73]. Jackson also provided a partially confirmed explanation of why this discrepancy existed: because of premature transcription termination in the leader region. The histidine workers also observed premature termination of transcription.

Jackson's work involved isolation and characterization of a variety of *trp* operon mutants. These mutants formed an important basis for further investigations by K. Bertrand. At the same time, T. Platt investigated ribosome-binding sites within the *trp* system, F. Lee worked on in vitro studies of attenuation, Craig Squires sequenced parts of the operon, Catherine Squires studied the interaction of the *trp*-repressor protein, RNA polymerase, and the *trp* promoter; and L. Korn investigated termination of transcription by polarity.

State Transformations

[PlattSY76] — Discovery of Leader-Region Ribosome-Binding Sites

[PlattSY76] describes experiments designed to find ribosome-binding sites in the vicinity of the leader region. Platt incubated *trp*-mRNA and ribosomes together in the absence of the remaining translational machinery, then added an enzyme that digests mRNA, and fingerprinted the remaining mRNA. The ribosome protected the ribosome-binding sites from digestion.

The authors expected to find only one ribosome-binding site, at the start of *trpE*. However, they found an additional unexpected site at the start of the leader region. They thought immediately that this site must somehow be involved in regulation of transcription termination at the attenuator. Perhaps the polypeptide from this region interacted with the attenuator.

Or perhaps regulation was accomplished via the physical presence of the ribosome itself, or by the act of translation.

These hypotheses were advanced with some reservations: no leader polypeptide had been detected, and other researchers had observed sites that attracted ribosomes, but that did not actually initiate translation — this site could have been an evolutionary relic from the *trpE* gene that no longer had a function.

[PlattY75] — Overlapping Translation-Termination and Ribosome-Binding Sites

[PlattY75] describes a study of the remaining ribosome-binding sites in the operon — those for the downstream genes. The investigators observed that the ribosome-binding site for *trpA* overlaps the end of the *trpB* gene. Such a juxtaposition had never before been observed in bacteria, so this observation was of general relevance to molecular genetics. It was not clear to Yanofsky's group at the time whether the overlap was relevant to regulation of the operon.

[SquiresLY75] — Investigation of Polymerase/Repressor/Promoter Interactions

[SquiresLY75] reported an in vitro investigation of the precise interactions among RNA polymerase, the *trp* promoter, and the *trp*-repressor. The work was concerned with questions such as the following: Do polymerase and repressor exclude each other from their binding sites? Can bound polymerase be dislodged from the promoter by the *trp*-repressor? In a control experiment, RNA polymerase, *trp* DNA, *trp*-repressor and *trp* were added to an in vitro transcription system, and normal repression of transcription was observed.

In another experiment, RNA polymerase was incubated with *trp* DNA for varying periods of time, then was added, with *trp* and *trp*-repressor, to the in vitro transcription system. The authors observed that, the longer the incubation period, the less repression occurred, suggesting that incubating polymerase and DNA produced resistance to repression.

Further experiments showed that (1) longer incubation times conferred resistance to repression by binding polymerase to more of the multiple copies of *trp* DNA present, not by increasing the strength of binding; (2) *trp*-repressor and RNA polymerase could not bind to the *trp* leader

simultaneously; and (3) trp-repressor inhibits transcription by preventing polymerase binding, rather than by preventing transcription by already-bound polymerase.

From these experiments the researchers concluded that there was a functional overlap of the promoter and operator.

[LeeSSY76] — Study of in vitro Transcription Termination

[LeeSSY76] describes an in vitro study of termination of transcription at the attenuator. These experiments used fingerprinting methods to analyze the composition of terminated trp-mRNA strands.

An initial experiment compared mRNA strands synthesized in vivo and in vitro, and showed that in vivo trp operon mRNA strands, where premature termination had not occurred, contained the in vitro strands on the 5' end, thus confirming that the in vitro strands were from the trp operon.

Subsequent experiments explored the kinetics of in vitro trp-mRNA synthesis. The experimenters halted transcription after increasing intervals of time had elapsed, and fingerprinted the mRNA that had been produced. In this way, the researchers were able to observe the leader region growing slowly, but never elongating past the attenuator.

The experimenters concluded that in vitro termination did occur at the attenuator, and that a very high proportion of transcripts terminate, because no full trp-mRNA transcripts were detected in vitro. They noted that they were unable to show whether RNA polymerase was actually dissociating from the DNA, or was pausing forever at the attenuator.

This paper also references experiments by Pannekoek, who observed a greater amount of in vitro transcription read-through than in these experiments, when Pannekoek added a protein he had isolated to the reaction mixture. Pannekoek postulated that this protein factor relieved attenuation. This "attenuation factor" was later found to be RNA polymerase.

[BertrandY76] — Attenuation is Regulated by Trp

The experiments described in [BertrandY76] showed that termination of transcription at

the attenuator is regulated by trp concentration. Trp-mRNA levels from cells in trp-rich media were compared levels from cells starved of trp. (The experimenters produced a state of trp starvation in cells by adding a trp analog to the media that both inhibited synthesis of trp by the biosynthetic pathway, and also competed with trp for binding to trp-repressor, but when bound did not activate the repressor.)

These studies compared levels of leader-region mRNA and structural-gene mRNA, and showed that the proportion of structural-gene mRNA was much higher in trp-starved cells.

Other measurements showed that starving cells of isoleucine (another amino acid) did not relieve termination, nor did any of the polypeptide products of the trp operon genes. The latter result contradicted previous observations by other researchers.

[BertrandSY76] — Additional Analysis of Leader-Deletion Mutants

[BertrandSY76] built on the work described in [JacksonY73]. The authors repeated some of Jackson's experiments on leader deletion mutants and performed a finer analysis of their results, and he performed several new experiments. This paper offers detailed evidence that premature termination of transcription occurs in vivo in the leader region.

The authors showed that the leader deletions do not affect the efficiency of the trp promoter. In addition, recombination experiments in mutants with elevated operon expression rates showed that the deletions were located near the operator-distal end of the leader region.

Merodiploid analysis (involving the addition of a plasmid containing a normal trp-operon leader region to mutant cells) showed that leader-region deletions are *cis*-dominant, indicating that these deletions were not inactivating a diffusable element such as a regulatory protein.

Another set of experiments distinguished among possible premature transcription termination, increased transcription initiation, and polymerase pausing by comparing relative transcription rates at two points within the trp operon: leader-region mRNA and *trpB* mRNA. The observation of differing rates of synthesis in different regions of the operon suggested that premature termination was occurring. A number of control experiments were performed to

strengthen this evidence. For example, the authors knew from earlier work that the different *trp* structural genes are expressed approximately equally, so premature termination is not occurring within the structural genes. The experimenters also verified that they were in fact measuring leader and *trpB* mRNA, and that differential hybridization rates were not skewing estimates of the relative transcription rates.

[SquiresLBSBY76] — Sequencing of Leader Region *trp*-mRNA

[SquiresLBSBY76] is largely a techniques paper that describes how mRNA from the leader region was sequenced by the analysis of mRNA fragments produced by digestion of the *trp* operon mRNA by several RNase enzymes. This work produced sequence data more accurate than those collected in the earlier work by [CohenYY73].

Obtaining the sequence allowed the authors to identify a translation-initiation codon for the leader region, as well as a downstream stop codon that was inphase with the initiation codon. They also observed regions of dyad symmetry within the leader region, and used a computer program to scan for stable RNA secondary structures. They found none, probably because the programs that searched for secondary structures were not very sophisticated at that time, and because there were errors in the sequence data that the authors analyzed.

[KornY76] — Interaction of rho with the Effects of Polarity and Attenuation

[KornY76] examined the interaction of the rho protein with the effects of polarity and attenuation. Polarity is a mechanism whereby the cell forces RNA polymerase to terminate transcription if polymerase is producing long transcripts that are not undergoing simultaneous translation by a ribosome (see Section 2.1.2).

These experiments showed that the rho protein was responsible for polarity by showing that polarity was suppressed in rho mutants. In addition, these *rho*⁻ strains showed reduced transcription termination at the attenuator; thus the authors concluded that rho is involved in attenuation. However, later experiments showed that in fact rho is not involved in attenuation, and that the *rho*⁻ strains used in these experiments actually contained a second mutation

called *trpX* that gave rise to the decrease in attenuation. Polarity is not involved in regulation of the *trp* operon, but is of general biological interest.

4.4.5 State 4: A Mechanism for Attenuation — Role of tRNA^{trp} and Secondary Structure

Description of Beginning of State

The knowledge at the start of this state is summarized in [BertrandKLPSSY75]. This review was published in 1975 and refers mainly to the papers described within state 3, which were still in press. The review indicated that the following points had been established:

- Deletion of a DNA site in the *trp*-operon leader region gave rise to elevated levels of *trp*-operon expression [BertrandSY76]
- The leader-region site caused premature termination of transcription in vivo [BertrandSY76] and in vitro [LeeSSY76]
- The degree of in vivo transcription termination is *trp*-sensitive [BertrandY76]
- Ribosome-binding sites for the *trp* structural genes, and one within the leader region, had been located [PlattSY76,PlattY75]

At this point, no one had determined exactly how the attenuator terminates transcription, or how the attenuator responds to changes in *trp* concentration. A number of possible mechanisms were proposed at the end of the paper: *trp* may affect the attenuator directly, or may affect the attenuator when bound to some other molecule; *trp* may inhibit a positive transcription read-through factor; or some metabolic product related to *trp* may influence the attenuator. Experimenters had postulated the existence of a leader peptide, and that it was involved in regulation of attenuation.

State Transformations

[MorseM76] — Attenuator is Sensitive to tRNA^{trp}

The studies described in [MorseM76] examined the role of tRNA^{trp} in the regulation of the trp operon. First the authors verified a previous result that tRNA^{trp} is not involved in repressor-mediated regulation of the operon.

Next, they verified that in *trpR*− strains, trp still regulated expression of the operon. They then measured operon expression when a *trpR*− mutant grew in the presence of two different trp analogs. One analog could serve as a corepressor, but could not charge tRNA^{trp}; it had no effect on expression of the operon in these (*trpR*−) mutants. The other analog could not be a corepressor, but could charge tRNA^{trp}; it reduced expression of the operon just as trp did. Performing the experiments in a *trpR*− mutant that also had its attenuator region deleted removed the effect mediated by the second analog. These experiments indicate that the attenuator region is sensitive to the charging of tRNA^{trp}.

The scientists noted that there are three specific mechanisms by which the attenuator might respond to tRNA^{trp}. It might respond to charged tRNA^{trp} as a negative control factor, or to uncharged tRNA^{trp} as a positive control factor, or to both. They postulated that palindromic sequences in tRNA^{trp} might bind to palindromic sequences in the leader to effect regulatory control.

[YanofskyS77] — Refinement of Understanding of Role of tRNA^{trp}

[YanofskyS77] describes another series of experiments that probed the interaction of tRNA^{trp} with the attenuator.

The investigators isolated a tRNA^{trp} mutant called *trpT_{ts}* that did not charge normally with trp. A strain containing this mutant and a *trpR*− mutation had elevated operon-expression levels compared to wild type, even in the presence of trp. If experimental conditions were altered to promote charging of this mutant tRNA^{trp}, expression decreased toward wild-type levels.

Yanofsky's group also isolated trp-tRNA-synthetase mutants, which they found increased trp-operon expression. In mutants with the attenuator deleted, this effect did not occur.

The conclusion drawn from this work was that the trp operon was sensitive to the concentration of either charged or uncharged tRNA^{trp}, but not to trp itself. The authors speculated that one of the tRNA species bound to a hypothetical attenuator protein and that this complex either increased termination at the attenuator, or increased read-through. They also concluded that this protein is not trp-tRNA-synthetase (Ito and his colleagues had proposed that trp-tRNA-synthetase acted as a repressor that was activated by charged tRNA^{trp} [ItoHY69], which pattern of repression has since been observed for another amino acid).

Another mutant, called *trpX*, was isolated in the course of these experiments. This mutant modified tRNA^{trp} in a manner that affected the interaction of tRNA^{trp} with the attenuator, altering the rate of trp-operon expression in *trpX* mutants. The experimenters determined that the mutants studied in [KornY76] in fact contained both a rho mutation and *trpX*, whereas Korn and Yanofsky had been unaware of the presence of *trpX* in 1976. Yanofsky and Soll determined that rho did not interact with the attenuator, in contrast to the conclusions of [KornY76].

[LeeY77] — Comparison of *E. Coli* Transcription Termination with that of *S. Typhimurium*

[LeeY77] describes comparative in vitro transcription experiments on fragments of the *E. Coli* and *S. typhimurium* trp operons. These fragments contained the leader promoter and leader regions of these operons plus a small amount of surrounding DNA.

Transcription of these fragments showed that some transcripts were terminated in the leader region, whereas other transcripts were terminated past the leader region. The fractions of these latter *read-through* transcripts were 5 percent in *E. coli* and 30 percent in *S. typhimurium*. The researchers anticipated that the difference in read-through percentages could be correlated with structural differences in the leader regions of the two operons.

A likely candidate for a structure that might vary is the secondary structure of the trp-mRNA. To search for secondary structures, the authors treated the mRNA with an enzyme that digests all single-stranded bases but leaves double-stranded regions (secondary structures) intact. They found a double-stranded stretch of mRNA toward the end of the leader region in both species. Examination of the DNA sequence of the leader region revealed two possible secondary-structure loops in both species. These structures — which were shaped like hairpins — shared a region of mRNA and thus were mutually exclusive: When one structure formed it prevented formation of the other. Sequence analysis predicted that the *S. typhimurium* hairpins would be less stable (correlating with its higher rate of read-through).

In another experiment, free GTP in the in vitro transcription system was replaced with the analog ITP. GTP is the chemical form in which the base G is found in solution. ITP is a chemical analog of GTP that does not base pair with C as strongly as it does with G, so when ITP is incorporated into the leader region, the stability of the hairpins should be reduced. The experimenters observed that attenuation was relieved in this experiment, which strongly suggested that the hairpin they found was involved in the premature termination of transcription.

They proposed that the formation of one or the other secondary structure determined whether or not transcription termination would occur. And they postulated that one or the other secondary structure forms based on whether or not a ribosome stalls at a leader-region trp codon for lack of a charged tRNA^{trp}. This hypothesis was supported by the previous experiments with tRNA^{trp}. Since the theory suggested that attenuation would be relieved whenever the ribosome stalled at a codon in the leader region, and codons existed in the leader region for amino acids other than trp, the researchers performed additional experiments to determine whether attenuation is relieved when the cell was starved for these other amino acids — such as isoleucine and arginine. Starvation for these amino acids did *not* relieve attenuation — an observation that conflicted with the predictions of the theory. (Later work

showed that the reason for the latter result is that isoleucine is not in the right place in the leader peptide to exert this effect. Although arginine is in the right place, this effect cannot be observed through measurement of enzyme production, because arginine is a component of the trp enzymes, and hence arginine starvation limits the production of these enzymes. Later experiments measuring trp-mRNA production rather than trp-enzyme production showed that arginine starvation does relieve attenuation.)

Of interest, a third mRNA stem and loop structure was found in the leader region near the other two. This structure was not reported in this paper because it was not necessary to explain attenuation, and the authors judged that it added an unnecessary complication [Yanofsky87]. The structure was later incorporated into the theory as a result of the analysis in [OxenderZY79].

These experiments were highly influential. They provided the missing link needed to generate a plausible theory of the mechanism of attenuation. The new theory incorporated a number of previously isolated pieces of this puzzle, such as the sensitivity of attenuation to tRNA^{trp}, the coding region in the leader discovered by Platt with its two tandem trp codons, and Jackson's experiments that showed attenuation was a *cis*-effect — that it was unlikely that the polypeptide product of the leader region was regulating attenuation.

[MiozzariY78a] — Comparison of *E. Coli* Leader Hairpins with those of *S. Marcescens*

[MiozzariY78a] presents studies of the trp operon in the bacterium *S. Marcescens*. In vitro transcription studies showed that transcription termination does occur in this leader region, suggesting that attenuation also occurs in this operon. Using sequence and hybridization data, the investigators looked for DNA regions that were conserved between the two organisms, under the assumption that such regions have an important function. Although some differences were observed, highly conserved leader hairpin-loop regions also were observed, and the promoter regions were found to be similar. One of the differences observed was that there are two

translation-start codons in the *S. marcescens* leader region.

The authors used newly developed restriction enzyme techniques extensively in these studies.

[ZurawskiBKY78] — The Phenylalanine Operon

[ZurawskiBKY78] describes studies of the *E. coli* phenylalanine (another amino acid) operon. The researchers sequenced leader-region mRNA from the phe operon and found a ribosome-binding site, a translation-start codon, an in-phase translation-stop codon (suggesting a leader peptide is formed by this operon), a 3'-OH mRNA terminus (suggesting that transcription termination does occur in the leader), and secondary structures that were similar to the trp leader stem structures. In addition, seven of the 15 codons in the hypothetical phe leader peptide coded for phe residues. This last observation was termed "startling!"

[OxenderZY79] — Details of the Leader-Region mRNA Secondary Structures

[OxenderZY79] describes studies that were similar to but more detailed than the experiments reported in [LeeY77]. Oxender and his colleagues also subjected leader-region mRNA to RNase digestion to search for secondary structures. These experiments, however, were able to show not only where within the leader-region base pairing was occurring, but what specific regions of mRNA were paired to what other regions. The authors of [LeeY77] had relied more on sequence data to postulate what specific structures formed.

Figure 4.7 shows the alternative secondary structures that this study found. Experiments indicated that structures 1:2 and 3:4 formed; theoretical calculations suggested that structure 2:3 would also be energetically stable. Figure 4.3 illustrates the essential aspects of the proposed mechanism of attenuation. When the cell is starved for trp, the ribosome stalls at the trp codons before region 2 due to a shortage of charged tRNA^{trp}. Thus, structure 2:3 is able to form, which prevents structure 3:4 (the termination signal) from forming. When the cell contains excess trp, the ribosome can read past the trp codons to cover region 2, preventing it from binding region 3 and thus allowing 3:4 to form.

The 1:2 structure found by Lee and Yanofsky but not mentioned in [LeeY77] was incorporated in the theory at this point. This structure is necessary to explain why termination in the attenuator occurs *in vitro* when translation is not occurring simultaneously. Structure 1:2 forms before 2:3, thus allowing the terminator structure (3:4) to form.

Data from earlier experiments by Yanofsky's group provided evidence for this model. Mutations that de-stabilized the 3:4 structure caused decreased termination at the attenuator, whereas mutations that destabilized the 2:3 (antiterminator) structure decreased the relief of attenuation at low *trp* concentrations. In addition, sequence data from other operons and other species showed similar structures in the leader region.

[BennettY78] — Precise Mapping of the *Trp* Operator

[BennettY78] mapped the location of the *trp* operator precisely using two functional assays. First, *trp*-repressor was incubated with *trp*-operon DNA and was allowed to bind to the DNA. The DNA was then digested and the resulting fragments were analyzed to see whether the repressor had protected certain restriction-enzyme sites in this vicinity. The repressor did protect a site between -9 and -14 (these numbers indicate distances in nucleotides downstream of the start of transcription), but did not protect a site between -32 and -37.

Second, five different operator-constitutive mutants were sequenced. All mutations were in the region -16 to -6.

[BrownBLSY78] — Localization of *Trp* Promoter Function

The research described in [BrownBLSY78] located the *trp* promoter based on two types of functional tests. A number of restriction fragments from this vicinity of the operon were obtained. For each fragment, two questions were asked: Could RNA polymerase transcribe this fragment? Could incubation with RNA polymerase protect this fragment from nuclease digestion?

These tests showed that polymerase did protect regions from -38/-41 to +18. It could not initiate transcription of a restriction fragment that began at position -39; if the cut was

made at -78, however, transcription did occur. So the promoter boundary appeared to be near -39. The authors obtained other results in *S. typhimurium*, consistent with the work of other researchers.

[BennettSBSY78] — Nucleotide Sequence of Promoter–Operator Region

The series of experiments described in [BennettSBSY78] yielded the nucleotide sequence of the promoter–operator region of the *E. coli* *trp* operon. Both the DNA and the transcribed RNA were sequenced.

[MiozzariY78b] — Translation of a Leader-Peptide Gene Fusion

[MiozzariY78b] demonstrated that the leader-region ribosome-binding site could in fact promote protein translation in vitro. Although the leader peptide itself could not be detected, in this experiment a leader-region deletion was used to fuse the DNA coding for the leader peptide to the *trpE* gene. The cell did transcribe and translate this fused gene to produce an observable fused polypeptide.

The scientists postulated that the leader peptide itself had not been detected because either it was degraded extremely rapidly, or the mRNA secondary structures detected by [LeeY77] prevented its synthesis.

4.4.6 State 5: The Attenuation Mechanism Is Confirmed and Refined

Description of Beginning of State

- [MorseM76] showed that the expression of the *trp* operon was sensitive to the concentration of charged tRNA^{trp}, and that this effect is distinct from repression. [YanofskyS77] confirmed these results, and suggested that expression was sensitive to both charged and uncharged tRNA^{trp}.
- Several studies concentrated on the operator–promoter region. Its sequence was obtained, and the extents of these sites were mapped using several functional tests.

- The leader region itself was studied closely — in isolation, in comparison to the leader region of the *trp* operon of another bacterium, and in comparison to the *phe* operon of *E. coli*. Sequence analysis showed the presence of alternative secondary structures in the leader region; Lee and his colleagues proposed that the formation of one or the other secondary structure determined whether transcription termination occurred at the attenuator. They postulated that selection between the secondary structures was influenced by whether a translating ribosome stalled at the *trp* codons in the leader region.

State Transformations

[StroynowskiY82] — Transcript Secondary Structures Do Affect Attenuation

The set of experiments described in [StroynowskiY82] examined the role of leader-region mRNA secondary structures in attenuation in the *S. marcescens* *trp* operon. A different bacterium was used because it included restriction-enzyme sites that allowed the experimenters to construct desired deletion mutants. *Trp* operon synthesis was studied in several mutants that included different leader-region deletions. All deletions started at the translation-start codon and extended varying distances in the 3' direction. The following effects were observed:

- Deleting only the translation-start codon gave increased termination (since the terminator could always form)
- Deleting region 1 (see Figure 4.7) increased expression (since the antiterminator always formed)
- Deleting regions 1 and 2 gave superattenuation, showing that secondary structure 3:4 alone could cause termination
- Deleting from the translation-start codon through regions 3 or 4 gave high expression (since the terminator was deleted)

Some deletions did not give the initially expected effects, but further analysis of the leader-region sequence revealed that new secondary structures could form that would interfere with the expected effects to produce the effects that were observed.

[DasCY82] — The Trp Operon Can be Regulated in vitro by Attenuation

A series of experiments showed that attenuation could be observed in an in vitro system [DasCY82]. One experiment involved adding a cell extract from a mutant with a temperature-sensitive trp-tRNA-synthetase to the in vitro system. The resulting expression of the trp operon was higher than it was when this mixture also contained a wild type trp-tRNA-synthetase; the latter produced greater quantities of charged tRNA^{trp}, thus increasing attenuation.

Yanofsky notes [Yanofsky87] that it is still unclear why charged and uncharged tRNA^{trp} compete, since the theory of ribosome action predicts that uncharged tRNAs do not interact with ribosomes.

[ZurawskiY80] — Conversion of Mutants with Low Operon Expression to Ones with High Expression

[ZurawskiY80] describes a series of experiments that began with trp-operon mutants that had lower-than-normal expression of the operon, and selected for new mutants with higher expression. The DNA sequence of the new mutants was analyzed, revealing that although some of the new mutants were simply revertants, most destabilized the 3:4 structure, suggesting that this structure is responsible for transcription termination. This experiment confirmed that mutations that interfered with regulation of the termination signal could be overridden by mutations that removed the termination signal.

[Platt81] — Generalizing Transcription Termination

Platt generalized the process of transcription termination from a review of this phenomenon in a variety of organisms and operons [Platt81]. For example, the sequences of 30 different transcription-termination regions were compared to produce a description of a typical termination region. In addition, rho-dependent termination, polarity, and antitermination factors

were discussed.

[KelleyY82] — Regulation of Trp-Repressor Production

[KelleyY82] studied the regulation of the *trp*-repressor protein. This repressor protein is coded for by a single gene. The gene is within an operon whose operator region binds the *trp*-repressor protein. Thus, the rate of synthesis of the repressor is sensitive to the concentration of *trp*. The biologists did not expect this operon to be self-regulating (*autogenous*), since the cell requires stronger repression of the *trp* operon (and presumably, more *trp*-repressor) at *high* *trp* concentrations, whereas an autogenous *trpR* operon should produce more *trp*-repressor at *low* *trp* concentrations (since at low *trp* concentrations, the *trp*-repressor does not bind to the operator of the *trpR* operon, and thus the operon is turned on).

Data from this study led to the hypothesis that, when the concentration of *trp* is high, *trp* saturates the small amount of repressor protein present to activate essentially all of that protein. In this state the *trpR* operon is completely shut down. Biologists know that at relatively low *trp* concentrations, the *trp* operon is still strongly repressed. Because the concentration of *trp* is low, little activated *trp*-repressor would be formed (contradicting the observation that the *trp* operon is strongly repressed) unless the concentration of the *trp*-repressor protein were high. Thus, the *trpR* operon synthesizes more *trp*-repressor at a high rate until enough repressor has been made to shut down both the *trp* operon and the *trpR* operon.

[DasUWNY83] — In vitro Synthesis of Leader Peptides

In the experiments described in [DasUWNY83], the researchers were at last able to synthesize the leader peptide in vitro. They accomplished this by inserting a small segment of the leader-region DNA, containing only the leader peptide, into a plasmid. When translation of a longer segment of the leader region was attempted, expression of the leader region was decreased by a factor of 10. Analysis of the leader-region sequence revealed complementarity between the leader-region ribosome-binding site and a distal portion of the leader region. The authors proposed that after the attenuation decision was made, these two segments paired to

mask the ribosome-binding site and to prevent further translation of the leader peptide, thus conserving the cell's resources.

[Dekel-GorodetskySE86] — Detection of the Leader Peptide in vivo

[Dekel-GorodetskySE86] demonstrated that a copy of the leader peptide that was inserted into a plasmid without the complementary distal segment could be translated in vivo.

4.5 Analysis

The chronology in Section 4.4 provided a detailed account of how the theory of the trp system changed over time. It tells us what the scientists knew at different points in time. But it does not tell us how each state of knowledge was derived from its predecessor. An analysis of the conceptual reconstruction has yielded a number of general patterns and characteristics of the trp research, which has led us to extract general strategies and principles for scientific-theory formation. More specifically, we shall discuss

- Strategies for experimentation that suggest when to undertake different types of experimental inquiry. These types of inquiry are called *modes of exploration*.
- A set of *theory-modification operators* that were used to generate new theories from old ones.
- Possible reasons why attenuation was correctly characterized by researchers working on the trp operon but not by researchers working on the his operon.

The analysis in this section is summarized in Figure 4.3 and in Appendix C. Appendix C lists, with the name of each paper, a short description of what phenomena that research unit investigated, and what changes to the theory of the trp operon resulted from the unit. Each paper in Appendix C is labelled with the mnemonic of the modification operators and modes of exploration that were applied in that unit of research. We claim that these operators and modes are sufficient to account for the evolution of the theory of the trp operon.

Figure 4.3 shows a state-space representation of the trp research in finer detail than the five gross states listed in Section 4.4, but in less detail than the paper summaries in that section. This diagram shows how knowledge of the trp system evolved over time and emphasizes the search space of alternative theories that the researchers explored. This figure labels different states of knowledge (A through J), and groups different papers into these states. Papers that are within boxes led off the main path of research because they presented hypotheses that were later invalidated. Finally, Figure 4.3 labels each state with one or more modes of exploration.

4.5.1 Modes of Scientific Exploration

One natural question to ask after reading a large number of scientific publications is this: why did the scientists perform those experiments in the first place? What principles would lead a machine to make similar choices? Examination of the experiments in the conceptual reconstruction reveals several different patterns of experimental inquiry. There are five different associations between the type of experiment performed at a given time and the state of knowledge of the trp operon at that time, which correspond to five different *modes of scientific exploration*. A mode of exploration is a pattern of experimental inquiry. When employing a mode of exploration, a researcher is guided to perform certain types of experiments. Different types of experiments are intended to affect, in different ways, the theories the researcher is considering. In addition to identifying the different modes of exploration, we have identified possible *transitions* between the modes. These transitions dictate what mode can be employed next, after work under a previous mode has terminated. The modes and the allowed transitions are shown in Figure 4.8, and are described in detail in the following sections. We assert that these modes of exploration are sufficient to account for why different types of experiments were performed at different times during the trp-system research. This claim is substantiated by Appendix C, in which every research unit has been assigned to one or more modes.

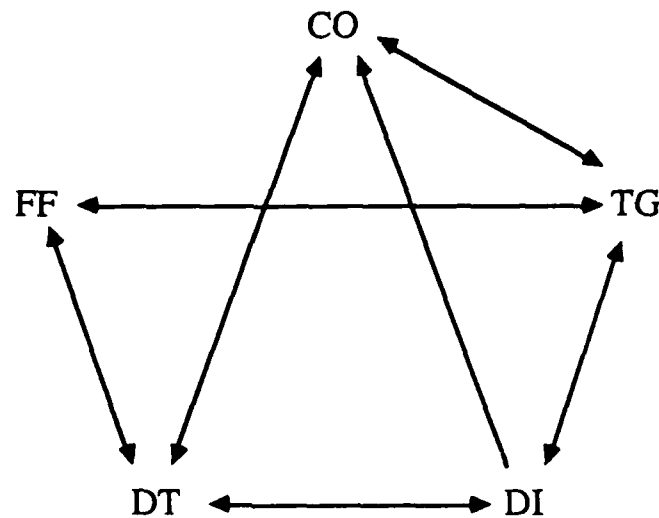


Figure 4.8: This figure shows different modes of scientific exploration and the allowable transitions between these modes. The modes of exploration are nodes of the graph; directed edges of the graph represent allowed transitions among the modes. CO = confirmation, TG = theory generation, DI = discrimination, FF = fact finding, TD = technique development.

Confirmation (CO)

The confirmation mode of exploration is invoked when researchers possess a single theory to which they have assigned a high degree of belief. Both theory- and data-driven experiments are employed within this mode. Theory-driven experiments are generated to test the implications of a theory with the intent of revealing potential flaws in that theory. Data-driven experiments are generated to refine the existing theory — perhaps in a slightly different physical system — with the chance that the theory will be disconfirmed due to an unexpected observation. The data-driven aspect of confirmation mode seeks to ensure that a theory does not blind a researcher to new phenomena.

One means of testing a theory is to determine whether phenomena that have been observed within the one or more physical systems within which the theory was developed, can also be observed within a different but similar physical system. For example, [Hiraga69] successfully generated and characterized trp-operator and trp-repressor mutants whose existence was predicted by the Jacob–Monod theory of repression (which was originally developed within the lac operon). [MiozzariY78b] applied the theory of attenuation to the trp operon of a different

bacterium: *S. marcescens*. [ZurawskiBKY78] applied the attenuation theory to a different operon of *E. coli* — the phenylalanine operon.

An important criterion in undertaking most research programs is the likelihood that they will lead to novelty. Since the Jacob–Monod theory had been applied to only one operon previously (the lac operon), it was both important to test the theory in a new system, and reasonably probable that something novel would turn up in the new system. What if five of the 20 total amino-acid operons had already been characterized? What if 10 had been? What if 19 had been? How does a scientist estimate how worthwhile it will be to devote resources to the characterization of a system that is analogous to other systems that have already been characterized? Some estimate of what will be learned from a given study must be made. This estimate must be based on the number of similar systems that have already been characterized, the amount of variation that has been observed in those systems, the existence of preliminary information that indicates that the system may not be behaving within an existing theory, and the relevance of the system at hand to clinical problems or to other research problems. Estimates must also be made of what experimental techniques are applicable to the system and of how quickly these techniques can yield new information.

In another class of CO experiments, scientists attempt to observe phenomena that the theory predicts should exist, but that have not been observed previously. For example, [LeeSSY76] searched for the short mRNA molecules whose existence was predicted by the theory of premature transcription termination advanced in [JacksonY73].

This mode also involves experiments to refine a theory by increasing the precision of its predictions. For example, [ForchhammerJY72] measured the rate of mRNA degradation by the cell and established the mechanism of this degradation. [CohenYY73] partially established the sequence of leader-region trp mRNA.

I call this mode *confirmation* because historically many experiments within this mode did in fact support the theory being tested. In general, the theory being tested was the

best available at the time. On occasion, the theory was disconfirmed by experiments within this mode. Popper might prefer to call this mode *refutation*, given his hypothesis that the fundamental mode of scientific inquiry involves subjecting scientific beliefs to an onslaught of tests in an attempt to refute these beliefs [Popper65]. The degree to which a theory withstands refutation is the degree to which it is corroborated. That attempts at refutation are not the sole goal of this mode is demonstrated by the numerous data-driven experiments also performed in this mode. These experiments were designed not to test the implications of the theory, but rather to measure some aspect of the system under study in more detail, such as in [ForchhammerJY72,BakerY72]. These more detailed measurements also may render a theory internally inconsistent, thus contributing to its refutation.

When a theory is refuted, the CO mode is abandoned. Refutation can result from experimental evidence that is inconsistent with the predictions of the theory. For example, the increased operon expression observed in some leader-region deletion mutants in [JacksonY73] was not predicted within the existing theory of repression. Refutation of the one existing theory under consideration leaves the researchers with no plausible theories to consider, so a transition occurs to the theory generation (TG) mode of exploration.

The decision as to whether a conflict between prediction and observation refutes a theory is not trivial. Both the accuracy and the precision of predictions and observations may be uncertain, so the fact that predictions do not match observations exactly does not necessarily refute a theory. In addition, the accuracy and precision of a theory may vary depending on how close to the domain of applicability that the theory is applied. It seems likely that scientists use knowledge of these factors to determine the severity of a prediction-observation mismatch. Another term for the mismatch severity is the *interestingness* of a phenomenon. I have not investigated the concept of interestingness in significant detail.

The CO mode can also be abandoned when researchers require new experimental techniques to probe a theory, in which case a transition to the technique-development (TD) mode occurs.

Theory Generation (TG)

The theory-generation (TG) mode of exploration involves the generation of new theories to describe the system under study rather than the design and execution of laboratory experiments. Theory generation occurs in [JacksonY73] (several theories were generated to account for the increased *trp* operon expression in the leader-region deletion mutants), [Kasai74] (where a more detailed but incorrect account of premature transcription termination in the leader region is proposed), and [LeeY77] (which formulated a theory of how alternative mRNA secondary structures regulate attenuation).

The theory-generation mode of exploration produces zero or more theories that are probed further within other modes of exploration. The net result of this process is not a set of “randomly” generated theories, but rather theories that have some reasonable degree of plausibility with respect to existing experimental data. For example, [JacksonY73] generated several plausible theories to account for the behavior of the leader-region deletion mutants they studied. One such theory involved premature transcription termination in the leader region; another involved decreased degradation of *trp* mRNA.

The term *generation* is not meant to imply that these theories are constructed from naught or in a vacuum. Theory generation is a highly knowledge-intensive process that is based, for example, on previously held theories of the system under study.

This mode is abandoned when no theories can be generated, in which case the fact finding (FF) mode is entered. If a single theory is generated, there is a transition to the confirmation mode. And if more than one theory is generated the discriminate (DI) mode is pursued. Section 4.5.2 discusses in more detail how the biologists performed theory generation.

Discrimination (DI)

The discrimination (DI) mode of exploration is important when several plausible theories exist to describe the observations at hand. It is used to discriminate among these theories with

the goal of increasing the difference in the credibilities of the theories until the credibility of one theory dominates the others. If this goal is accomplished a transition is made to the confirmation mode, where new experiments are performed in the context of the dominant theory. It is possible that experiments performed in discrimination mode will refute *all* theories under consideration. In this case, a transition to theory-generation mode occurs. It is also possible that existing experimental techniques will be unable to measure the quantities that would discriminate among several theories, in which case a transition to TD mode occurs.

A number of strategies are employed in this highly theory-driven mode of exploration. Experimenters may attempt to provide positive or negative support for a single theory by observing phenomena that are consistent with the predictions of the theory. Alternatively, the predictions of two or more theories could be compared until some testable difference between their predictions can be deduced, at which point an experimental test is performed — if possible — to observe which of the conflicting predictions of the theory is confirmed.

Part of the work in [JacksonY73] occurred in discrimination mode: A number of experiments were performed to discriminate among the theories that were generated to explain the high trp-operon expression of the leader-region deletion mutants. [BertrandSY76] reported similar work in which many of Jackson's experiments were repeated in more detail, and new experiments were performed to discriminate among theories that Jackson had not considered.

Fact Finding (FF)

The fact-finding (FF) mode is important when no plausible theories to explain the system under study can be generated. Perhaps a large number of implausible theories can be generated, but performing experiments to select among these candidate theories would require a prodigious amount of time. Thus, a general goal of fact-finding mode is to obtain more knowledge that will be of use in constraining the generation of plausible theories. In the trp system, this general goal was often satisfied by one or more experiments that initially had little probability

of achieving that goal. Experiments were often undertaken in a fairly haphazard way, such as by applying a newly developed experimental technique to the system under study (e.g., DNA sequencing). The term *haphazard* is not meant to imply any lack of competence; all studies labelled as FF produced scientific results of publishable quality. Rather, the term indicates that at the time the experiment was formulated, there was no way to tell whether or how the experiment would contribute to the solution of the more general theory-generation problem at hand. These experiments are highly data-driven and are designed to produce data that are interesting in their own right, as opposed to data that should be relevant to particular theoretical anomalies.

Examples of fact-finding mode occur in state 3. After [JacksonY73] determined that premature termination of transcription was occurring in the leader region, no particularly good theories were generated to provide a molecular mechanism for this process, or to explain its sensitivity to the concentration of trp as elucidated in [BertrandY76]. Thus, a set of fairly uncoordinated experiments were undertaken: [PlattY75,PlattSY76] searched for ribosome binding sites within the trp-operon leader region and structural genes; [SquiresLBSBY76] sequenced the trp-mRNA leader region; [KornY76] studied the interaction of the rho protein (involved in transcription termination) with expression of the trp operon; and [SquiresLY75] explored the detailed in vitro interactions of RNA polymerase, trp repressor, and the trp promoter and operator. All these experiments were of general biological interest and investigated important phenomena whose details were not well established. They applied newly developed biological techniques, such as DNA sequencing, to the trp operon to characterize it in novel ways. None of these experiments was explicitly designed to reveal the mechanism of attenuation. However, it happened that several of the experiments, in conjunction with [LeeY77], yielded the mechanism for attenuation. These experimental results were integrated synergistically and fortuitously to yield a theory that no direct approach had produced.

When new knowledge is derived from FF-mode experiments, a transition is made to theory-generation mode to determine whether the new knowledge permits the generation of plausible theories. If existing experimental techniques are not sufficient to observe the entities that researchers wish to measure, a transition occurs to the TD mode. This transition occurred in [CohenYY73] — Cohen and his colleagues developed new RNA-sequencing techniques to establish the sequence of the trp-operon leader region.

As noted in Figure 4.8, FF-mode experiments occurred not only when the researchers were unable to generate a plausible theory to account for some anomaly, but also during the entire course of the research. The work described in [BennettY78,BrownBLSY78,BennettSBSY78] and [NicholsVY81] was all FF mode. Each study explored a property of the trp operon that was not expected to be directly relevant to the understanding of attenuation, at a time when other members of the laboratory were operating within different modes of exploration. Probably, these FF experiments were valuable for several other reasons:

- When a scientist pursues a line of research, it is always possible that her efforts will lead one down a dead-end trail. A large investment in research could produce an incorrect theory that bears little resemblance to an accurate theory of the system under study. FF-mode experiments provide alternate potential avenues of research as well as the potential for indirect checks on the main thrust of the research. They are thus a way of hedging one's bets.
- Serendipity plays an important role in science. Broadening a program of research may increase the likelihood that an unexpected but important effect or anomaly will be observed.
- Many different people worked in Yanofsky's laboratory over the years of this research. They had diverse scientific interests, from the mechanism of mRNA degradation to the general structural properties of proteins. Yanofsky permitted his students to work on

any scientific question they chose, as long as it involved *trp* or the *trp* operon in some way [Yanofsky87]. Thus, many FF-mode experiments resulted from the diverse interests of students — an example is Platt's work [PlattY75,PlattSY76].

- Many of these studies produced publishable scientific results in diverse areas that were of value in and of themselves. Some of these results were expected (for example, that the sequence of the leader region would be obtained in [CohenYY73]), and some were not (such as the overlapping start codon and ribosome-binding site found by [PlattY75]).

Technique Development (TD)

In the TD mode of exploration, researchers develop new experimental techniques that are more efficient (for example, less costly or less time consuming) than are existing techniques, or that are able to measure aspects of a biological system that were previously unobservable.

For example, Cohen and his colleagues developed new techniques that allowed them to sequence longer regions of mRNA more quickly than previous techniques had allowed [CohenYY73]. They used their new method to sequence the leader region of the *trp* operon. Other techniques developed by Yanofsky's group included methods for observing transcription termination in vitro [LeeSSY76] and in vivo [BertrandSY76], and methods to prepare leader-region deletion mutants [JacksonY73].

The *trp* researchers also employed techniques that recently had been developed by other molecular biologists, such as the DNA-sequencing method developed in the mid-1970s, and the method for locating ribosome-binding sites used by Platt.

Summary of Modes of Exploration

The determination of which mode of exploration to employ at a given time usually is correlated with both the number of competing theories currently under consideration and the degree of credibility assigned to each of these theories. The major difference between the modes

of exploration is the degree to which the experiments they advocate are theory-driven, as opposed to data-driven. Making this distinction precise is not a trivial pursuit, because almost all experiments seek to acquire data to benefit theory. The distinction refers to the degree to which data are sought in order to extend a theory's domain of applicability or to increase the precision of its predictions, as opposed to in order to confirm a theory or to decide among competing theories. Data-driven experiments seek to refine a theory by observing those aspects of a system that the theory does not predict. That is, data-driven experiments seek to measure some aspect of the system about which the theory makes little or no commitment, such as the sequence of a gene. Theory-driven experiments generally seek to make a measurement whose outcome has been predicted by a theory, in an attempt to test the validity of that theory.

Consider the following problems with these definitions. Data-driven experiments are driven by theory in that it is necessary to examine a theory in order to determine what attributes of the system are not predicted by that theory. So data-driven experiments are driven by holes in the theory. And all data-driven experiments take place within the context of a theory that can be confirmed or disconfirmed by their results; but data-driven experiments are not designed to test a theory directly or to decide among competing theories. Similarly, theory-driven experiments do yield useful new data that often are not predicted by the theory. This framework is appealing in that its theory-driven modes are employed when the currently held theory or theories are reasonably credible, and its data-driven mode is employed when they are not.

Comparison with the Hypothetico-Deductive Model

It is instructive to compare the modes of exploration with the classic hypothetico-deductive model of science developed by philosophers of science [Hempel66]. This model contains three different modes of exploration in which a scientist (1) generates a theory, (2) deduces consequences of the theory, and (3) performs experiments to test the predictions from step 2. If

the prediction is incorrect, the theory is falsified, so the scientist retreats to step 1. If the prediction is correct, more rounds of testing occur through use of steps 2 and 3.

The major differences between the hypothetico-deductive model and the preceding modes of exploration are that the latter includes fact-finding mode, which provides for the times when theory generation does not generate only a single theory, but rather generates zero theories or more than one theory. When zero theories are produced, fact finding mode is entered to provide more data to aid theory generation. When multiple theories are produced, discrimination mode is entered to discriminate among them. The hypothetico-deductive model lacks any mechanism for producing the less directed, serendipitous observations of fact finding mode, whose usefulness was discussed earlier. It also lacks certain aspects of confirmation mode. Work in confirmation mode can be data-driven, as when researchers explore a new physical system by making general measurements of the system. Although this work does take place in the context of an existing theory, it often is not directly designed to test predictions of the theory, as in steps 2 and 3. The hypothetico-deductive method also lacks a technique-development mode.

The Model-System Model of Research

The research on the trp system is an instance of a general pattern of biological research called *research on model systems*. (This mode of exploration occurs at a higher level than those discussed earlier.) This pattern involves the investigation of a number of different phenomena (such as bacterial gene regulation) through intensive study of a single biological system that exhibits those phenomena: the model system. Examples of model systems are the trp operon, the lac operon (in which the theory of repression was developed), and the lambda phage (in which an understanding of the regulation of viral genes was developed). These systems are "models" in that they are by far the most well understood examples of phenomena that appear in many biological systems.

The most distinctive aspect of this type of research is that a diverse set of research projects is undertaken within the model system. For example, within the trp system, Platt searched for ribosome-binding sites within the trp operon to investigate general properties of the translation process [PlattY75,PlattSY76]. Bennett, Brown, and Squires investigated general properties of repression by mapping the precise location of the trp operator and promoter, and by probing repressor-operator-polymerase interactions [BennettY78,BrownBLSY78,BennettSBSY78]. Often, experiments in the model system unexpectedly lead to new lines of inquiry. For example, Das's attempts to express the leader peptide in vitro led him to discover the new mechanism by which translation of the leader peptide is regulated [DasUWNY83].

There are several possible alternatives to the model-system approach. Instead of utilizing a common biological system, researchers might choose to investigate a single phenomenon in many different systems; for example, a group might study attenuation in many different bacterial operons. Or, a group might prefer little direction, allowing its members to pursue whatever topics they find interesting in whatever systems they prefer to study. As noted earlier, these types of synergy occurred in state 3, when Yanofsky pieced together results from a number of projects in his laboratory to determine the mechanism behind attenuation [BertrandKLPSSY75].

There is one general reason that research in a model system can be more efficient than are its alternatives: There is a potential for synergistic interactions among individual research projects. These interactions comprise sharing experimental techniques and sharing experimental data.

Even though different researchers may be investigating diverse biological topics, their work may require many of the same experimental apparatuses and techniques, because they are working within the same model system. For example, assays for the trp enzymes were used in many experiments, and some mutant *E. coli* strains were used in several different projects. It is more efficient to share this type of experimental expertise than to develop new techniques

from scratch for different biological systems.

Often, data from earlier experiments in the trp system were employed in some newer theory-formation task, although the earlier experiments were not performed with the latter task in mind. The earlier data were relevant by chance. For example, the sequence of the trp operon has been used in many different projects, few of which were explicitly anticipated at the time the trp operon was sequenced. Such data can be invaluable for testing newly formed theories, and may save researchers large amounts of time.

4.5.2 Theory Generation

To gain insight into the process of theory generation, we compared the theories of the trp system that existed before and after each unit of research. By studying the differences between theories of the trp system that existed at different points in time, we can note regularities in the types of changes that are made to biological theories. Generalizing from these changes allows us to postulate a set of theory-modification operators. Each operator takes one theory as input, and as output produces one or more new theories that differ from the input theory in specific ways. (This notion is refined in Chapter 5). The main points of this section are as follows:

- The theory-modification operators employed by biologists include all possible syntactic operations that can be applied to the components of scientific theories that were described in Chapter 3
- Generally, each unit of research produced only one or two theory modifications of this sort

Evidence for these claims is presented in Appendix C, which summarizes each unit of research by briefly describing what phenomena the researchers studied, and what new knowledge their experiments added to the theory of the trp system. In addition, for each unit of research,

we considered how the theory it produced would be represented using the framework in Chapter 3. We then considered the syntactic difference between the new theory and the theory that existed before that unit of research. Appendix 3 summarizes these syntactic modifications in the line labelled "Changes" for each paper. For example, [JacksonY73] proposes a new interaction between objects in the trp system — namely that the attenuator region causes termination of transcription. We can represent this new interaction by adding a new process to the trp theory; thus, the entry for [JacksonY73] is annotated with "CP" (create process). Figures 4.5 and 4.6 show the studies in which one or more new theories of the trp operon were generated. These diagrams show the space of alternative theories that the researchers generated and explored.

The operators themselves are described in more detail in the following sections. Here, we make general observations about the data in Appendix C.

The majority of the papers make only one or two syntactic changes to the theory. Thus, we generally see limited, conservative modifications to a theory within a single unit of research. For example, [Imamoto70] measures the time lag required for repression to take effect after trp is added to a culture; [MorseM76] adds a process describing the influence of tRNA^{trp} on attenuation.

A majority of the units of research studies altered the credibility of the theory in some way.

The theory-formation operators can be divided into five classes, depending on which syntactic aspect of the theory they modify. These classes are for modification of processes, objects, system state variables, degree of belief, and domain of applicability. In the following sections we discuss each of these classes in more detail.

Object Modifications (NO)

Several studies resulted in the addition of a new object to the theory of the trp system. That is, the theory asserted the existence of an object that had not been recognized previously.

[PlattSY76] determined that a ribosome-binding site existed within the leader region. [Kasai74] asserted the existence of a protein factor that bound to RNA polymerase during transcription initiation and that prevented premature transcription termination in the *his* leader region. Both [JacksonY73] and [Kasai74] postulated the existence of an attenuator site within the leader regions of the *trp* and *his* operons, whose behavior was to cause premature transcription termination.

The new objects that were introduced into theories bore different degrees of similarity to previously known objects. Some new objects were straightforward instantiations of known classes of objects. For example, the ribosome-binding sites discovered by Platt [PlattSY76,PlattY75] were not significantly different from any other known ribosome-binding site. Thus, this object was simply instantiated from an existing class of objects (we use the notation `NO:I` to denote `new object:instantiate`).

In contrast, the protein and the DNA sites proposed in [Kasai74] and in [JacksonY73] were quite different from other known proteins and DNA sites. That is, their behaviors were novel: No class of proteins was known to bind to RNA polymerase and to prevent transcription termination. No class of DNA sites was known to cause transcription termination at a frequency that depended on the cellular concentration of *trp* (or of any other chemical). These objects, nonetheless, did bear similarities to previously known entities. Both were constructed from previously known chemicals: protein and DNA. Both displayed behaviors that were generally similar to those of previously known proteins and DNA sites. Many proteins bind to objects within the cell and alter the behavior of that object. In fact, the rho protein was then known to cause (rather than to prevent) termination of transcription. Also, the researchers already knew of a DNA site that caused termination of transcription independent of *trp* concentration: the transcription terminator. Thus, some properties of the new objects were present in existing objects, but some properties were specially tailored so that the addition of these objects would endow the theory with the desired predictions. These new objects were instantiated not from

an existing class, but from a new class of objects that was postulated to exist (we call this action NO:P, for new object: postulate).

Finally, in some cases the properties of existing objects were refined (NO:R), as in [CohenYY73], where the sequence of the 5' end of trp-mRNA was determined. Similarly, [BennettY78,BrownBLSY78] determined the precise location of the trp operator and promoter, respectively.

Process Modifications (MP, CP)

Another class of changes made to the theory of the trp system involved assertions about the behaviors of both new and existing classes of objects. As the theory of the trp system evolved, the researchers proposed that previously known types of objects could interact in previously unknown ways, and they proposed behaviors for newly created objects. Within the GENSIM framework, we would propose such behaviors either by modifying the definitions of existing processes (MP), or by creating new processes (CP). (We never observed the removal of a process from the theory; in general, however, such removal appears to be a reasonable type of theory revision.)

[BertrandY76] asserted that trp regulated the rate of transcription termination at the attenuator. [KornY76] postulated an interaction between the rho protein and the attenuator. [MorseM76] asserted that the attenuator was sensitive to the presence of tRNA^{trp}. [LeeY77] postulated that RNA polymerase interacts in two different ways with the alternate mRNA secondary structures that form in the leader region.

Parameter Refinements (PR)

Frequently, a study obtained a value for a state variable of the trp system more precise than the previously known value. For example, [ForchhammerJY72] quantified the degradation rate of trp-mRNA, and [RoseY72] measured the rate at which trp-mRNA was synthesized in different

media.

Extension/Restriction of Domain of Applicability (EX/RE)

Biologists intend that every biological theory be used for predicting the behavior of some portions of the physical world, and not for predicting that of others. We refer to this region of a theory's competence as its *domain of applicability*.

Many units of research changed the trp theory's domain of applicability. Usually the domain was extended, but on occasion it was restricted. [MiozzariY78b] extended the trp theory to apply to the trp operon of the bacterium *S. marcescens* in addition to that of *E. coli*. [ZurawskiBKY78] extended the theory to apply to the phenylalanine operon of *E. coli* in addition to the *E. coli* trp operon. Most restrictions imposed on a theory were due to anomalous findings that demonstrated that it did not produce valid predictions under all circumstances. For example, [JacksonY73] showed that the existing theory did not make correct predictions when applied to trp-operon mutants with deletions in their leader regions.

Increase/Decrease in Belief in a Theory ($\Delta B + / \Delta B -$)

A large fraction of the studies produced results that influenced the confidence the researchers placed in the predictions of the theory. As we would expect, findings in agreement with predictions of the theory (particularly predictions that were not tested previously) increased the degree of belief in the theory, whereas inconsistent findings decreased confidence in the theory. For example, [Hiraga69] reported an early study that applied the Jacob-Monod repression theory to the trp operon. Since the researchers observed anticipated classes of operator and repressor mutants, their confidence that the Jacob-Monod theory applied to the trp operon increased. When Baker and Yanofsky studied the rate of transcription initiation in the trp operon, however, they observed a higher than expected rate of transcription initiation immediately after wild-type cells were placed in a trp-free medium [BakerY72]. This finding decreased

their confidence that the Jacob–Monod theory was valid for the trp operon.

There is an interesting correspondence between the degree of belief in a theory and that theory's domain of applicability. In some cases, we can define degree of belief in terms of domain of applicability since often our belief in a theory decreases as the domains to which that theory is applied diverge from the one in which it was developed.

4.5.3 The Histidine Operon

This section compares research on the trp operon with a similar program of research on the *E. coli* his operon. Given that the researchers investigating these operons were exploring similar systems, but that only one group successfully elucidated the mechanism of attenuation, it is instructive to consider why one group was successful, whereas the other was not.

Recall from our discussion of state 2 that the his researchers proposed a mechanism for attenuation in which uncharged tRNA^{his} allows a protein factor to bind to RNA polymerase at the promoter; RNA polymerase could continue transcription past the attenuator only when bound to this protein (according to their theory). As we have seen, this proposed mechanism was largely incorrect, in that there is no such relief-protein — the attenuator forms alternate secondary structures based on whether or not a ribosome stalls in the leader region. However, we can view the ribosome as a positive factor here, and in fact the ribosome is probably the limiting factor that the titration experiments in [ArtzB75] revealed.

We have identified several possible reasons why the his researchers did not discover the correct mechanism of attenuation. First, they apparently did not generate a hypothesis involving ribosome stalling and mRNA secondary structures. Artz has confirmed that they were unable to generate a hypothesis that reflected the linkage they had observed between relief translation and transcription termination [Artz86]. Second, someone may have generated this hypothesis, but then rejected it because he could not imagine how the cell might select between

the alternative secondary structures. Yanofsky's group was also unable to postulate an acceptable molecular mechanism for attenuation after the experiments in [JacksonY73]. However, Yanofsky's group shifted into a fact-finding mode of exploration and accumulated enough new findings about the trp operon that they were soon able to establish such a mechanism.

Another difference between the work of the his researchers and Yanofsky's group is that the his researchers committed to the detailed protein-factor hypothesis relatively early in their work. They did not perform analogs of the control experiments used in [JacksonY73], or of the broad set of experiments performed in state 3, to verify that premature termination of transcription was occurring in the manner they proposed. That is, they did not enter a confirmation mode of exploration.

4.6 Summary

We have studied a series of discoveries in molecular biology using techniques from knowledge engineering and information-processing psychology. The first phase of this study yielded a detailed conceptual reconstruction of the states of knowledge through which the biologists progressed as their understanding of the trp operon increased. The development of the theory of the trp operon resembles the evolution of a species. As it was subjected to different experimental pressures, the theory took on a number of competing forms; most forms were abandoned in favor of a single emergent theory. Figures 4.5 and 4.6 diagram this evolutionary process.

In the second phase of the study, we searched for principles of scientific-theory formation sufficient to generate the states of knowledge elucidated in the first phase. The biologists used five modes of exploration to determine what type of experiment to perform next given the current state of knowledge:

1. Confirmation
2. Theory generation

3. Discrimination

4. Fact finding

5. Technique development

The selection of a mode of exploration is usually determined by the number of alternative theories in the current state of knowledge, and by the degree of credibility assigned to these theories. The experiments generated by different modes of exploration usually differ in the degree to which they are data- versus theory-driven. As an example, discrimination mode generates theory-driven experiments, and is employed when several alternative theories that have approximately equal credibilities exist. We also discussed the strategy of directing a large research group to investigate many different facets of a single biological system — a model system. This approach is efficient because it allows researchers to share experimental techniques and data.

We also sought to understand the principles used to generate new scientific theories. By comparing the differences between consecutive states of knowledge of the trp operon, we were able to identify several general types of modifications that were made to theories. The predictions of the theory contained either more or less precision, were assigned more or less credibility, and applied to either an extended or a restricted domain. These changes in the behavior of a theory were usually accomplished through the following types of changes to the content of a theory: determination of a more precise value for a parameter of the theory, determination of the internal structure of an object in the theory, introduction of a new object into a theory, or introduction of a new interaction between objects into the theory. Note that these principles do not indicate *when* different operators should be applied, but only establish a set of operators that can be applied. This question is addressed in Chapter 5.

Finally, we compared the evolution of the attenuation theory advanced by the biologists who studied the *E. coli* histidine operon, to the evolution of the trp operon theory. We identified

several possible reasons why the his workers did not derive the correct theory. It appears that they did not pursue the appropriate modes of exploration, and that they were unable or unwilling to generate and pursue the correct theory.

Chapter 5

Hypothesis Formation by Design

This chapter presents methods for improving the predictive power of a scientific theory given new information obtained by experiment. These methods are meant to be employed when a scientist performs a laboratory experiment whose outcome is not correctly predicted by existing theory. At such times, scientists often form one or more hypotheses that improve their ability to predict the outcomes of both the current experiment, and of experiments performed previously in that scientific field.

A program called HYPGENE embodies these methods for hypothesis formation. HYPGENE solves hypothesis-formation problems from the history of attenuation presented in Chapter 4. Its hypotheses improve the predictive power of theories expressed within the GENSIM framework (described in Section 3.5). This chapter describes the methods used by HYPGENE; Chapter 6 describes the implementation of HYPGENE and compares its methods to those used by previous researchers. Chapter 7 describes hypothesis-formation problems that HYPGENE has solved.

This chapter addresses a subset of the overall problem of hypothesis formation. Section 5.1 states the overall problem, and then describes the subset of the problem addressed here. This thesis treats hypothesis generation as a *design* problem. Section 5.2 provides an overview of

this design metaphor and of HYPGENE's methods. Section 5.3 presents an annotated solution to a hypothesis-formation problem that was computed by HYPGENE. HYPGENE's inputs are expressed as *design goals*, which are described in Section 5.4. HYPGENE constructs hypotheses by modifying an existing theory using several types of *design operators*, which are discussed in Section 5.5. HYPGENE uses search methods to apply the design operators. Section 5.7 discusses the computational complexity of HYPGENE. Methods for controlling HYPGENE's search are described in Section 5.8. The most powerful and novel of these methods constrains hypothesis formation using information from a *reference experiment*.

5.1 The Hypothesis-Formation Problem

In 1973, Jackson and Yanofsky created *E. coli* mutants from which a region of the *trp* operon had been deleted [JacksonY73] (see Section 4.4.3). Their theory of the *trp* operon said that the deleted region of DNA had no function, so the rate of expression of the operon should be unaffected by this deletion. But they observed that the expression of the *trp* operon increased significantly.

A scientist takes several types of actions when the predictions of her theory are not consistent with her experimental observations. She may formulate new theories that predict the observations correctly, such as proposing that the deleted DNA region had a regulatory function. She may choose to discard certain experiments because she does not deem their results trustworthy — Jackson and Yanofsky might have decided that their experiments were contaminated. She may identify certain experiments as outside the theory's domain of applicability, and thus as irrelevant to testing the theory; for example, Jackson and Yanofsky might have decided that their anomalous observations had nothing to do with gene regulation, but were due to changes in enzyme function. Finally, she may postulate that the initial conditions or observed outcomes of certain experiments are different from what she originally thought

Notation	Meaning
E_A	An anomalous experiment
I_A	Initial conditions of E_A
P_A	Predicted outcome of E_A
O_A	Observed outcome of E_A
$Error_A$	Error in predicted outcome of E_A
E_R	A reference experiment
I_R	Initial conditions of E_R
P_R	Predicted outcome of E_R
O_R	Observed outcome of E_R

Table 5.1: Notation for describing experiments.

they were, in a way that allows her theory to predict their outcomes correctly — Jackson and Yanofsky might have interpreted their instrument readings in a new way that indicated that the expression of the operon did not in fact increase.

More abstractly, the general hypothesis-formation problem is as follows. We are given two inputs:

1. A set of experiments \mathcal{E} that is relevant to the theory at hand, where each experiment consists of a description of the starting state of a physical system, and the final observed state of that system
2. A theory T that is intended to be used for both predicting the outcomes of the experiments in \mathcal{E} , and evaluating the experiments in \mathcal{E}

Hypotheses must be generated because \mathcal{E} contains some *anomalous experiment* E_A whose outcome as predicted under T (called P_A) does not match its observed outcome, O_A . The theory T is used to predict the outcome of E_A from a description of its initial conditions, I_A (Table 5.1 summarizes this notation). Therefore, the following relationships hold:

$$I_A \cup T \models P_A$$

$$P_A \neq O_A$$

We must produce two outputs:

1. A set of new alternative theories \mathcal{T} that is able to correctly predict the outcomes of all experiments in the set \mathcal{E}_3 (see (2)). The set \mathcal{T} may have one or more members and may include T .
2. Three new sets of experiments, \mathcal{E}_1 through \mathcal{E}_3 , where
 - (a) \mathcal{E}_1 contains those elements of \mathcal{E} whose outcomes the theory does not need to predict correctly, because the scientist considers these to be bad experiments. For example, the scientist might conclude that the techniques used in these experiments are inaccurate, or were employed improperly.
 - (b) \mathcal{E}_2 contains those experiments in \mathcal{E} that the scientist considers trustworthy, but whose outcomes are not correctly predicted by any theory in \mathcal{T} ; these are problematic experiments for which the scientist is unable to generate an acceptable theory.
 - (c) \mathcal{E}_3 contains those experiments in \mathcal{E} whose outcomes can be correctly predicted. More precisely, \mathcal{E}_3 contains three subsets of experiments:
 - i. \mathcal{E}_{3A} contains experiments taken from \mathcal{E} exactly as they were performed originally. That is, for some T_i in \mathcal{T} and some experiment in \mathcal{E}_{3A} with initial conditions I_A :

$$I_A \cup T_i \models P_A' \qquad P_A' = O_A$$
 - ii. \mathcal{E}_{3B} contains experiments taken from \mathcal{E} but with modifications to their initial conditions. That is, for some experiment in \mathcal{E}_{3B} with modified initial conditions I_A' :

$$I_A' \cup T_i \models P_A' \qquad P_A' = O_A$$
 - iii. \mathcal{E}_{3C} contains experiments taken from \mathcal{E} but with modifications to their observed outcomes (O_A might be modified because a new interpretation is given to raw

data from the experiment).

$$I_A \cup T_i \models P_A \qquad P_A = O_A'$$

Some readers may find it odd to think of modifying the initial conditions of an experiment to align its predicted and observed outcomes. While studying the history of attenuation, however, we noted that biologists often modified their idea of what the initial conditions were because their knowledge of I_A was uncertain. One reason for the uncertainty is the incredible complexity of the objects in I_A — bacterial DNA is millions of bases in length, and biologists have only partially determined its structure. Another reason for the uncertainty is that experimental conditions are often tailored using laboratory techniques whose effects cannot be predicted with complete certainty, such as gene-splicing techniques.

The problem addressed in this dissertation is a subset of the preceding problem. I do not consider formulating hypotheses for an arbitrary set of experiments \mathcal{E} , but instead, for two cases: (1) the case where \mathcal{E} contains a single element E_A that is an anomalous experiment, and (2) the case where \mathcal{E} contains two elements: an anomalous experiment E_A , and a *reference experiment* E_R . A reference experiment is one whose initial conditions I_R are similar to those of the anomalous experiment (I_A), but whose predicted outcome *does* match its observed outcome:

$$I_R \cup T \models P_R \qquad P_R = O_R$$

Second, I do not address the problem of how to critique experimental techniques, so HYP-GENE does not generate either the set \mathcal{E}_1 , or \mathcal{E}_{3C} . I assume that all experiments are sound. I do consider how to generate the sets \mathcal{E}_{3A} and \mathcal{E}_{3B} , which involves modifying I_A and T .

More precisely, the problem considered here is, given

1. A theory T
2. An anomalous experiment E_A whose outcome is *not* correctly predicted by T , and possibly given a reference experiment E_R whose outcome is correctly predicted by T

We are asked to produce:

- A set of hypotheses, where each hypothesis is a tuple $\{T', I_A', I_R'\}$ such that the predicted outcomes of both experiments now match their observed outcomes.

$$I_A' \cup T' \models P_A'$$

$$P_A' = O_A$$

$$I_R' \cup T' \models P_R$$

$$P_R = O_R$$

Note that the outcome of E_R must be unchanged, since its predicted outcome was already correct. In addition, because we assume E_A and E_R are so similar, the same modification must be applied to both I_A and I_R :

$$I_A' - I_A = I_R' - I_R$$

There is a simple mapping between these modifications and the theory-representation framework presented in Chapter 3. Modifications to the initial conditions of an experiment involve modifications to the set of objects present in the simulation KB at the outset of an experiment, and to the properties of those objects. Generating new theories involves modifying the process knowledge base, including modifying and deleting existing processes and creating new processes.

It is possible to describe the difference between the predicted and observed outcomes of an experiment (the prediction error) in several different ways.¹ Figure 5.1 diagrams the prediction error, which can be thought of as all assertions contained by the prediction and the observation, minus the assertions that are common to both:

$$Error_A = (P_A \cup O_A) - (P_A \cap O_A)$$

¹Whereas GENSIM represents I_A and P_A as frames in a KEE knowledge base, HYPGENE represents them as sets of predicate-calculus assertions, and the latter representation facilitates the discussion that follows. Nilsson describes how to translate between frame and predicate-calculus representations [Nilsson80].

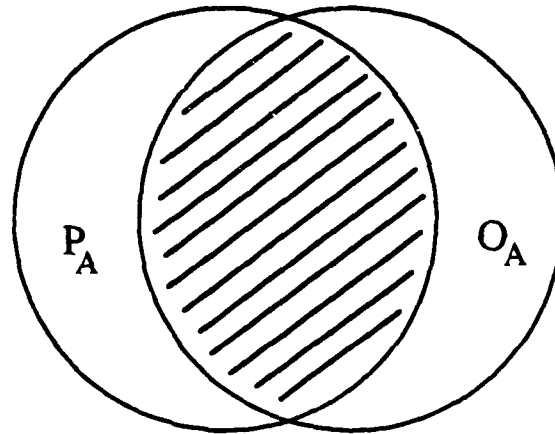


Figure 5.1: The error in a GENSIM prediction is the unshaded regions. O_A is represented by the right hand circle. One crescent represents the assertions that the prediction should have but lacks (the add list, AL_A), the other crescent represents assertions that the prediction includes, but should lack (the delete list, DL_A).

It can also be described as the assertions that are present in the prediction but should not be, plus the assertions that are present in the observation, but are missing from the prediction.

$$Error_A = (P_A - O_A) \cup (O_A - P_A)$$

The two terms in the preceding equation can be thought of as an *add list* (AL) and a *delete list* (DL). We can transform the set P_A into O_A by adding to P_A those assertions that O_A contains but P_A lacks, and by deleting those assertions that P_A contains but O_A lacks. Thus:

$$Error_A = AL_A \cup DL_A$$

$$O_A = P_A - DL_A \cup AL_A$$

This view of the prediction error is important because it says that virtually all prediction errors can be expressed as an add list plus a delete list. Most of HYPGENE's design operators reflect this view because they address the goal of either adding assertions to a prediction, or removing assertions from a prediction. The exceptions to this classification of prediction errors are quantitative prediction errors. Even though GENSIM does not compute quantitative

predictions, HYPGENE can compute quantitative hypotheses — hypotheses that explain why the amount of an object in a prediction is too high or too low. Such hypotheses do not completely remove a population of molecules from a prediction, nor create a population where it did not exist before; thus the assertions in a prediction are unchanged. Section 5.4.2 discusses quantitative hypotheses in more detail.

5.2 A Design System for Hypothesis Generation

This dissertation argues that it is beneficial to treat hypothesis formation as a design problem. Design is a metaphor for hypothesis formation in that it provides a general framework for thinking about hypothesis-formation problems. In addition, I show how methods developed by AI researchers to solve problems in design and planning can be used to solve hypothesis-formation problems. Although the process of design is not completely understood by AI, it is better understood than is the problem of hypothesis formation, thus making this approach profitable. The remainder of this section describes the process of design in general, shows how design is analogous to hypothesis formation, then provides an overview of the techniques employed by HYPGENE.

Design is a creative activity in which a designer constructs an artifact that satisfies a set of constraints. Traditionally, we think of designing tangible objects, such as digital circuits, bridges, and houses. We can also think of designing more abstract entities, such as a plan of action (a robot path plan, for example) or a computer program. Extending the concept even further, we can imagine designing a legal system or a programming language. All these entities perform some function, from spanning a waterway to providing a language for writing certain types of computer programs. Put another way, the behavior of these objects must satisfy certain constraints. In addition, all these entities are constructed from primitive components — the design primitives — such as suspension cables, transistors, and two-by-fours.

To treat hypothesis formation as a design problem, we view a theory as an artifact that scientists construct.² Theories function to predict the outcomes of experiments; thus, theories (and hypotheses) are designed subject to the constraint that their predictions must match experimental outcomes. In general, theories should satisfy other constraints as well: Their predictions should be testable, they should be consistent with other scientific knowledge, and they should satisfy certain syntactic requirements such as simplicity. To call a theory a designed artifact, we must identify the design primitives from which theories are constructed. Chapter 3 does just this — it describes a framework for defining theories in which the design primitives are frames in a class knowledge base, and a language for defining processes. So, when cast in this light, the hypothesis-formation problem is to design modifications to a theory and to the initial conditions of an experiment that render the predicted outcome of the experiment compatible with the observed outcome.

AI researchers have approached design problems using our central paradigm of search. The search space for a design problem is the space of all possible ways of combining the design primitives. Solution states are those arrangements of the design primitives that satisfy the design constraints. The search operators are design operators that combine the design primitives into larger arrangements. This is the approach used by HYPGENE, except that when designing theories, HYPGENE starts not with the bare design primitives, but rather with a partially correct theory that its design operators modify. Thus, it is more accurate to say that HYPGENE *redesigns* than it is to say HYPGENE designs.

In the discussion that follows, I will describe HYPGENE's design constraints and design operators, and methods that HYPGENE does and could use to control its search. The design metaphor suggests other methods that could be brought to bear on the hypothesis-formation problem, but that are not discussed in detail in this dissertation. Just as designers sometimes

²Unfortunately, scientists use the word *artifact* to refer to interesting experimental observations that were caused by technical errors in an experiment, rather than by the properties of a system under study; I use the word to mean a designed entity.

record the reasons for their decisions among alternative designs to facilitate the redesign of an artifact, a designer of theories could maintain a design history for a theory to facilitate modifications to that theory. Just as designers sometimes reason at different levels of abstraction, so too could the designer of a theory. Finally, many designers maintain libraries of existing designs so that new designs can be adapted quickly from old ones, rather than constructed laboriously from naught; HYPGENE does in fact modify a theory that it is given, but its starting library contains only a single theory.

HYPGENE's I/O behavior is shown in Figure 1.3. Its inputs are

- I_A — the initial conditions of an experiment
- P_A — the predicted outcome of the experiment computed by GENSIM
- $Error_A$ — the difference between P_A and the observed outcome of the experiment

HYPGENE's output is

- I_A' — a modified version of the initial conditions of the experiment

HYPGENE's design goal is to modify I_A such that there is no longer any difference between prediction and observation — such that the prediction error $Error_A$ is eliminated.

This chapter describes four classes of hypothesis-design operators:

- Initial-condition design operators
- Process-design operators
- Class KB design operators
- Quantitative-hypothesis design operators

Only the first and last classes of operators are implemented within HYPGENE. The design operators apply syntactic modifications to I_A , T , and the CKB. The initial-condition design

operators alter I_A ; for example, by adding objects to I_A , or by removing objects from I_A . The process-design operators modify processes in the PKB, for example, by generalizing the preconditions of a process. These operators are syntactically complete in that they can generate any initial conditions and any theory that can be represented using the GENSIM representation language. So, if these operators were applied to I_A and T at random, they would eventually arrive at solutions for I_A' and T' that satisfy the design goals.³

HYPGENE does not, however, apply its design operators randomly; it uses a form of means-ends analysis to direct the use of the operators [NewellS63]. Section 5.1 showed that a prediction error can be broken down into an add list of assertions to be added to P_A , and a delete list of assertions to be removed from P_A . All the initial-condition design operators and process-design operators are directed toward one of these two ends. HYPGENE works backward from its design constraints and attempts to apply operators that will satisfy particular design constraints. In addition, the operators are able to make use of information from GENSIM's simulation dependency trace, so that if the goal is to remove an object from P_A , and the object is present in the prediction because a process created it, then an operator will easily be able to determine what conditions caused that process to fire, and will try to violate those conditions.

Chapter 6 describes the implementation of HYPGENE in detail; here we give a brief outline of its architecture. HYPGENE performs a best-first search of its hypothesis space. An agenda mechanism assigns priorities to unexpanded search states. Each search state contains a state-specific description of the design goals, and of I_A' and P_A' . HYPGENE's execution cycle is shown in Figure 5.2.

HYPGENE is called a *designer* rather than a *planner* because traditionally the output of a planner is a temporal sequence of actions. The hypotheses generated by HYPGENE have no temporal component and thus are more similar to the class of artifacts synthesized by the process of design. The distinction between planning and design is fuzzy, however, because the

³A random search would terminate at about the same time that the monkeys at the word processors finish the second act of King Lear (assuming that the monkeys do not figure out how to use the spelling checker).

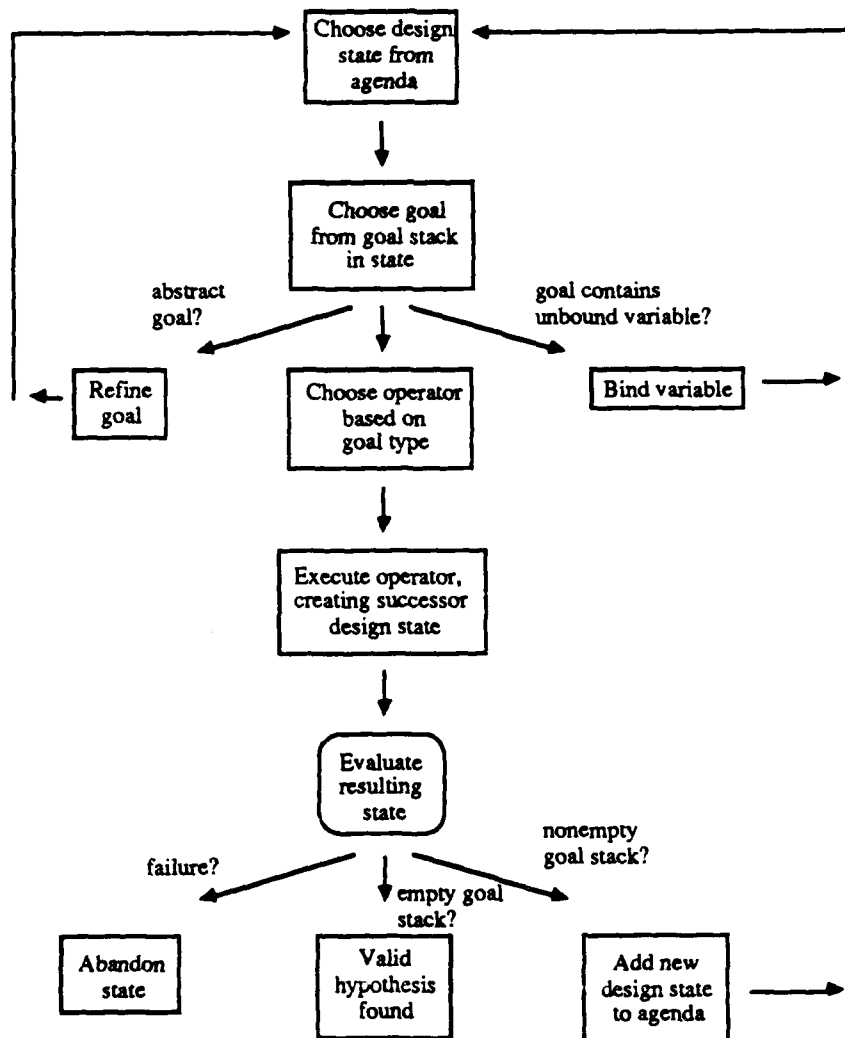


Figure 5.2: A flow chart of HYPGENE's execution cycle. HYPGENE executes these steps repeatedly until no design states remain on its agenda.

techniques the two employ are similar.

5.3 An Example Hypothesis-Formation Problem

Before describing the details of how HYPGENE works, we present an edited example of HYPGENE's reasoning. The biological experiment involved is shown in Figure 5.3. This experiment involves repression of the trp operon, and contains the following objects in its initial conditions: the trp operon, the trp aporepressor, and trp. GENSIM predicts that two reactions will occur: Trp binds to the aporepressor and activates it; the resulting complex then binds to the trp operator. Similar experiments were performed by [Hiraga69].

We tell HYPGENE that GENSIM's prediction is incorrect in that no repressor-operator complexes are observed, and HYPGENE formulates hypotheses to explain this error. The prediction error is described to HYPGENE as shown below. The first line indicates what variables are universally quantified and what variables are existentially quantified (variable names begin with "\$"). The expressions under "Outstanding constraints" form HYPGENE's current goal stack; these are predicate-calculus formulae that HYPGENE will attempt to satisfy. The predicates and functions in these expressions are explained in Table 3.1. The variable \$obj is universally quantified, so HYPGENE's starting goal is that no repressor-operator complexes should exist:

```
Uvars: ($obj) Evars: NIL
Outstanding constraints:
  (NOT (OBJECT.EXISTS $obj 'RepOp.Complexes))
```

In what follows we examine the problem-solving actions that HYPGENE takes and observe how its goal stack changes in response to these actions. This line of reasoning leads to one of the 15 solutions HYPGENE finds to this problem; the other solutions are summarized in Section 7.11.

HYPGENE first determines if its universally-quantified goal is satisfied, and finds that the

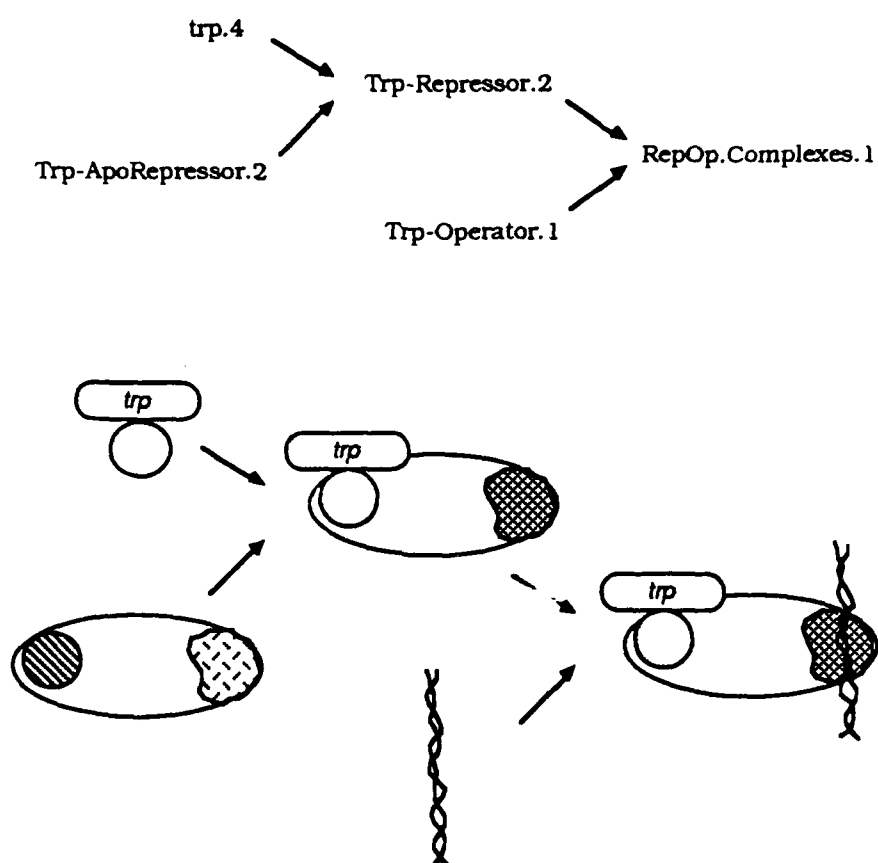


Figure 5.3: A sample GENSIM prediction. The top half of the figure shows the names of the reacting objects; the bottom half diagrams the objects themselves. A simple molecule called `trp` binds to a binding site within the `trp-aporepressor` protein. The protein is then in a form that allows its other binding site to attach to a region of DNA called the `trp operator`.

goal is violated by the object `RepOp.Complexes.1`. HYPGENE therefore creates a subgoal to remove the assertion that this object exists:

```
Outstanding constraints:
SUBGOAL.0316
  [(Remove.Assertion (OBJECT.EXISTS 'RepOp.Complexes.1
                                   'RepOp.Complexes])
```

The operator called `Remove.Assertion` consults the GENSIM simulation dependency trace and finds that `RepOp.Complexes.1` was created by a process. To delete this object, HYPGENE creates a subgoal to prevent the process that created `RepOp.Complexes.1` from firing.

```
Outstanding constraints:
SUBGOAL.0318
  ((Prevent.Prcs.From.Asserting
    (Trp-Repressor.Binds.Operator.PACT.1 XCINITEXPT)))
```

HYPGENE knows several ways to prevent a process from firing; one method is to violate one of the preconditions of the process. HYPGENE *refines* the `Prevent.Prcs.From.Asserting` goal to use this method:

```
Outstanding constraints:
  (Violate.Prcs.Condition.To.Prevent.Assertion
    (Trp-Repressor.Binds.Operator.PACT.1 XCINITEXPT))
```

HYPGENE next executes the operator `Violate.Prcs.Condition.To.Prevent.Assertion` on the process-activation record `Trp-Repressor.Binds.Operator.PACT.1`. The operator determines that the formula below expresses a condition that, if satisfied, will violate the preconditions of the `Trp-Repressor.Binds.Operator` process. It is not important to understand every clause within this goal. It is important to note that the formula is a disjunction and thus expresses a number of alternative ways of satisfying the goal, for example, by removing the `trp` operator (the first disjunct below). The disjunction was obtained by negating the preconditions of the process `Trp-Repressor.Binds.Operator`.

```
Outstanding constraints:
SUBGOAL.0338
  Uvars: ($Asite $site.interaction.class0329)
```

```

Evars: ($mutation0327 $obj0328)
[(OR (NOT (OBJECT.EXISTS 'Trp.Operator.1
           $site.interaction.class0329))
      [NOT (MEMB $site.interaction.class0329
                (GET.VALUES $Asite
                  'Potential.Interacting.Objects]
      (NOT (OBJECT.EXISTS $Asite 'Active.Sites))
      (NOT (IS.PART.R $Asite 'Trp-ApoRepressor.3))
      [AND (MEMB $obj0328
                (GET.VALUES $Asite
                  'Object.Interacting.With.Site))
      (OBJECT.EXISTS $obj0328
        (GET.VALUE $Asite
          'Potential.Interacting.Objects]
      (AND (IS.PART $mutation0327 $Asite)
        (OBJECT.EXISTS $mutation0327 'Mutations)
        (MEMB 'Trp-Repressor.Binds.Operator
          (GET.VALUES $mutation0327 'Processes.Disabled])

```

HYPGENE attempts to satisfy this disjunction by instantiating some of its universally-quantified variables and then considering each disjunct separately, since each disjunct represents a different way of satisfying the overall expression (Section 6.1.3 describes how HYPGENE satisfies quantified expressions). In what follows, HYPGENE considers the last disjunct, which involves a mutation in the site within the trp-repressor protein that binds to the trp operator. This disjunct says that an object in the class `Mutations` must exist, that the object must be part of the binding site in the trp-repressor, and that the specificity of the mutation must be such that it interferes with the process `Trp-Repressor.Binds.Operator` (the process that we are attempting to prevent from firing).

Outstanding constraints:

SUBGOAL.0352

```

Evars: ($mutation0327)
[(IS.PART $mutation0327 'Trp-R.Operator.Binding.Site.3)
 (OBJECT.EXISTS $mutation0327 'Mutations)
 (MEMB 'Trp-Repressor.Binds.Operator
   (GET.VALUES $mutation0327 'Processes.Disabled])

```

In order to satisfy this existentially quantified goal, HYPGENE first creates a new mutation object called `Mutations.17` and binds the variable `$mutation0327` to `Mutations.17`. This is depicted in Figure 5.4.

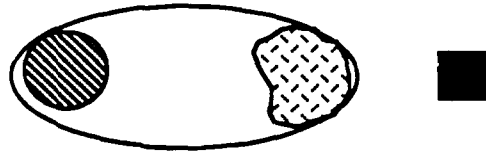


Figure 5.4: HYPGENE first creates a mutation object that is distinct from the trp-aporepressor protein.

Actions:

```
[[ASSERT (OBJECT.EXISTS 'Mutations.17 'Mutations]
```

Outstanding constraints:

SUBGOAL.0352

```
[[Add.Assertion (MEMB 'Trp-Repressor.Binds.Operator
                  (GET.VALUES 'Mutations.17
                              'Processes.Disabled]
```

```
(Add.Assertion (IS.PART 'Mutations.17
                    'Trp-R.Operator.Binding.Site.3]
```

The next condition that HYPGENE considers specifies that the `Processes.Disabled` slot of the `Mutations.17` object should have a certain value. Since the `Mutations.17` object is part of the initial conditions of the experiment, HYPGENE creates a subgoal to modify I_A to achieve this goal.

Outstanding constraints:

```
[Modify.Initial.Conditions.To.Add.Assertion
  (MEMB 'Trp-Repressor.Binds.Operator
    (GET.VALUES 'Mutations.17
                'Processes.Disabled]
```

The operator `Modify.Initial.Conditions.To.Add.Assertion` modifies I_A by adding the required value to the `Processes.Disabled` slot of `Mutations.17`:

Actions:

```
[[ASSERT (MEMB 'Trp-Repressor.Binds.Operator
              (GET.VALUES 'Mutations.17
                          'Processes.Disabled]
```

The remaining goal specifies that the `Mutations.17` object must be part of the binding site `Trp-R.Operator.Binding.Site.3`. However, this goal references an object (`Trp-R.Operator.Bindi`) that was created by a process; since this object was not present in I_A , HYPGENE cannot use

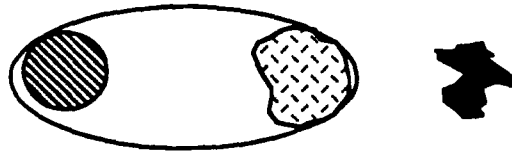


Figure 5.5: HYPGENE next modifies a slot in the mutation object that describes the specificity of the mutation.

the operator `Modify.Initial.Conditions.To.Add.Assertion` to modify it. Instead, HYPGENE posts a new subgoal to modify this binding site by modifying the inputs to the process that created it. That is, this operator attempts to modify the products of a reaction indirectly, by modifying the reactants.

```

Outstanding constraints:
  (Modify.Prcs.Input.To.Add.Assertion
    (IS.PART 'Mutations.17
      'Trp-R.Operator.Binding.Site.3))

```

The process `Trp-ApoRepressor.Binds.Trp` created `Trp-R.Operator.Binding.Site.3` by copying the object `Trp-R.Operator.Binding.Site.2`. HYPGENE posts a goal to modify the latter object on the assumption that the modification will be copied by the process. This modification changes the specificity of the mutation object, depicted as a change in shape in Figure 5.5.

```

Outstanding constraints:
  (Add.Assertion (IS.PART 'Mutations.17
    'Trp-R.Operator.Binding.Site.2))

```

The above goal is refined to a goal to modify I_A :

```

Outstanding constraints:
  (Modify.Initial.Conditions.To.Add.Assertion
    (IS.PART 'Mutations.17
      'Trp-R.Operator.Binding.Site.2))

```

The operator `Modify.Initial.Conditions.To.Add.Assertion` makes `Mutations.17` a component of

`Trp-R.Operator.Binding.Site.2`, as depicted in Figure 5.6. This action is different from

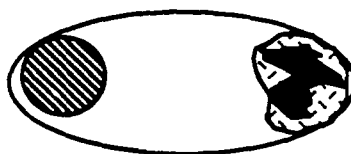


Figure 5.6: HYPGENE inserts the mutation into a binding site of the trp-aporepressor protein.

previous ones, because it modifies an object that participated in a reaction. The modified version of the object may or may not cause the same reactions. So, HYPGENE finds all processes that the original `Trp-R.Operator.Binding.Site.2` had caused to fire, and deletes all objects created by these processes (including `RepOp.Complexes.1`). HYPGENE then calls on GENSIM to recompute what processes the modified `Trp-R.Operator.Binding.Site.2` will fire (GENSIM computes the P_A' corresponding to the new I_A'). GENSIM finds that the modified repressor protein can still bind trp, but cannot bind the trp operator because of the mutation in `Trp-R.Operator.Binding.Site.2`. HYPGENE's starting design goal is satisfied because in the new version of the experiment, only one of the two earlier reactions can occur, and no repressor-operator complex is created.

Actions:

```
[(ASSERT (IS.PART 'Mutations.17
                'Trp-R.Operator.Binding.Site.2)
  (Unfiring (Trp-ApoRepressor.Binds.Trp.PACT.1 XCINITEXPT))
  (Unfiring (Trp-Repressor.Binds.Operator.PACT.1 XCINITEXPT))
  (Process Trp-ApoRepressor.Binds.Trp created Trp-Repressor.22))
```

This particular line of reasoning by HYPGENE yields one solution to this hypothesis-formation problem. Section 7.2.2 describes the other solutions that HYPGENE finds to this problem.

5.4 The Design Goals

HYPGENE's design goals describe two types of differences between the predicted (P_A) and observed (O_A) outcomes of an experiment. The first type of prediction error was discussed in Section 5.1; it involves extra assertions in P_A , or assertions that are missing from P_A . These

assertions represent objects with specified properties that should be added to or removed from P_A . The second type of design goal requests HYPGENE to generate hypotheses that would alter the concentrations of existing objects in P_A .

5.4.1 Goals Involving Assertions

A user utilizes predicate-calculus formulae to represent design goals involving the addition or removal of assertions from P_A : existentially quantified formulae specify that some object with given properties should exist in P_A , and universally quantified goals specify that no objects with given properties should exist in P_A . Another way of representing the latter goals would be to explicitly list those assertions that should be removed from P_A , rather than using universal quantification to specify them indirectly. I prefer to use quantification because it guards against hypotheses that do remove the erroneous assertions from P_A , but have the side effect of creating the exact same objects through some other reaction. These other objects would have different names than those listed explicitly, but would match universally quantified goals, and thus HYPGENE would detect that such a hypothesis was not a solution.

A typical hypothesis-formation problem in this domain is: Measurements show that no operator-repressor complexes are observed in an experiment, although their existence is predicted; generate hypotheses to explain this observation. This problem would be encoded as the following HYPGENE design goal:

```
(FORALL $X (NOT (OBJECT.EXISTS $X 'RepOp.Complexes)))
```

5.4.2 Quantitative Design-Goals

As described in Chapter 3, GENSIM predictions do not contain a quantitative component — GENSIM does not predict the concentrations of the objects it creates in a simulation. But, if the user of HYPGENE has a separate quantitative theory that lets him predict the concentration of an object in a simulation, and if the user determines that this concentration differs from

the observed concentration of the object, then the user can employ HYPGENE to formulate hypotheses to account for this discrepancy. Quantitative design-goals direct HYPGENE to either increase or decrease the concentration of some object. Users specify these goals using two special predicates: `Increase.Quantity` and `Decrease.Quantity`.

For example, Jackson and Yanofsky performed an experiment in which the observed concentration of mRNA was higher than the concentration they predicted. HYPGENE can formulate hypotheses to explain this (and does in Section 7.2.3). The Jackson-Yanofsky anomaly is described to HYPGENE by using GENSIM to predict the outcome of the experiment (that is, what reactions occur and what objects are created), and then giving HYPGENE the following goal (where `Messenger.RNAs.15` is the mRNA in the GENSIM prediction that the user knows must be increased):

(`Increase.Quantity` 'Messenger.RNAs.15')

It is important to note that HYPGENE cannot use GENSIM to simulate quantitative hypotheses, whereas HYPGENE does call on GENSIM to simulate and verify hypotheses that modify the assertions in I_A (as described on page 224).

5.5 Design Operators

HYPGENE satisfies design goals by employing *design operators*. The designer is able to examine its outstanding (as yet unsatisfied) goals, and to choose operators that will satisfy these goals. Section 5.1 showed that a prediction error can be broken down into an add list of assertions to be added to P_A , and a delete list of assertions to be removed from P_A . All the initial-condition design operators and process design operators are directed toward one of these two ends. The designer executes an operator by making changes to I_A , T , and the CKB, that the operator specifies. Operators often specify conditions that must be true before they can be executed; when HYPGENE selects an operator, it adds the operator preconditions to its goal

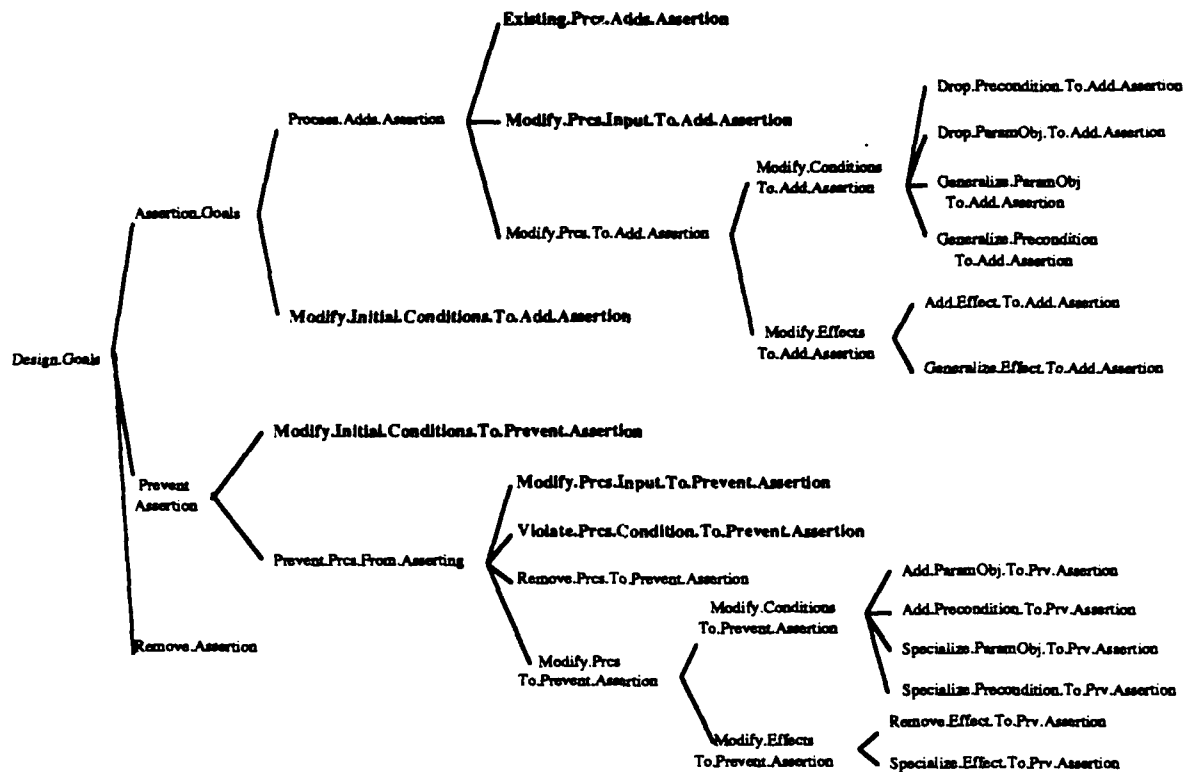


Figure 5.7: The hierarchy of initial condition design operators and process-design operators. The leaves in the tree are executable operators that achieve the goals to their left, for example, the operator **Add.Effect.To.Add.Assertion** modifies the effects of a process to satisfy the design goal of adding an assertion to P_A . Executable operators that modify I_A are leaves of the tree printed in boldface; process-design operators are leaves of the tree in normal type.

stack. A *valid hypothesis* results from a set of operator applications that satisfies all goals on HYPGENE's stack. Valid hypotheses are the outputs HYPGENE produces.

This section describes four different types of operators:

- Initial condition operators, which modify I_A
- Process modification operators, which modify T
- Quantity-hypothesis operators, which address goals related to the quantities of objects present in P_A
- Class KB operators, which modify the classes in the class KB

5.5.1 Initial Condition Design Operators

An assertion A could be present at the end of the simulation (and thus be an element of P_A) for only one of two reasons:

- Because A was present in I_A (the presence of A in P_A follows from the monotonicity of GENSIM simulations)
- Because a process fired (executed) and asserted A

Thus, one HYPGENE operator adds an assertion A to P_A by adding A to I_A ; several other operators modify I_A or T or both such that a process that asserts A fires in the current prediction. Preventing an assertion A from existing in P_A is essentially the converse of adding an assertion to P_A — A can be removed from I_A , or the process that asserted A can be prevented from firing. However, removing A from P_A is a bit more complex than adding A because the assertion could have been *both* present in I_A and asserted by one or more processes; all justifications of A must be eliminated if A is to be absent from P_A . Note that because object forking (see Section 3.5.6) requires that objects are never deleted from a simulation, it is impossible for the firing of a process to remove an assertion from P_A , so there is no operator that attempts to remove an assertion from P_A by firing a process.

Figure 5.7 shows HYPGENE's initial-condition design operators and process-design operators. Although the operators are implemented as LISP functions, they are organized conceptually in a class-subclass hierarchy that is used to control their execution (discussed in Section 6.1.3). Only leaf nodes of this tree are operators that can be executed during problem solving. We now describe each operator for modifying initial conditions in detail.

**Operators Modify.Initial.Conditions.To.Add.Assertion,
Modify.Initial.Conditions.To.Remove.Assertion**

These operators alter P_A by adding an assertion to the initial conditions I_A , or by removing an assertion from I_A , respectively. For example, the operator `Modify.Initial.Conditions.To.Remove.Assertion` would satisfy the goal

`(NOT (OBJECT.EXISTS 'trp.1 'trp))`

by deleting the object `trp.1` from I_A , if the operator found that `trp.1` existed in I_A .

Operator Existing.Prcs.Adds.Assertion:

This operator attempts to add an assertion A to P_A by searching the PKB for a process that, if executed, would assert A . For example, if the goal is

`(EXISTS $x (OBJECT.EXISTS $x 'RepOp.Complexes))`

then this operator finds all processes in the PKB that create an object of type `RepOp.Complexes`. Once the operator has found a process that can satisfy its goal, it must ensure that the process will fire within this prediction. To do so, it posts a new design goal on HYPGENE's goal stack that specifies the conditions that must be true for this process to fire: Objects of the types specified in the parameter-object classes of the process (which are explained in Section 3.5.3) must exist, and the preconditions of the process must be satisfied.

The task of finding a process with effects that satisfy a given goal is not as simple for HYPGENE as it is for STRIPS-like planners [FikesN71]. These planners encode the effects of their operators using an add list and a delete list. These lists specify assertions that are added to and deleted from working memory when an operator is executed. To find an operator that satisfies a particular goal, STRIPS-like planners match the goal against the add list of each operator. This task is conceptually simple because the goal and the add list are represented in

the same declarative language. Not so for HYPGENE. Although some processes do assert simple lists of propositions, the effects of other processes call LISP functions that perform complicated tasks, such as copying objects whose parts are arranged in arbitrary tree structures. These functions contain recursion and cannot be represented using fixed add and delete lists.

HYPGENE employs an incomplete solution to this problem. A preprocessor computes an add list for most processes (no delete list is needed because of simulation monotonicity) by simulating the execution of processes on typical parameter objects. For each process R , the preprocessor creates an object from each parameter-object class of R (in an otherwise empty experiment KB). Then the preprocessor executes R on these objects. The add list of R is created from an analysis of the objects created by R (after the execution of R has terminated).

The weakness of this method is that it assumes that the effects of R are always the same, because the add list it computes for R is based on a single set of prototypical parameter objects. Consequently, this method will work when either of the following conditions hold of a process R :

- The effects of R are not dependent on the properties of its parameter objects (such as their parts or slot values)
- The effects of R do depend on the properties of its parameter objects, but all possible such parameter objects can be enumerated by the preprocessor; an add list must be computed for each one

These conditions hold for most but not all of the processes in the trp system, so this operator cannot reason fully about some of the processes in the PKB.

Operator Violate.Prcs.To.Prevent.Assertion:

This operator is used to remove an assertion from P_A that was asserted by a process R . It does so by preventing R from firing. R fired because its parameter objects were present in the

SKB, and its preconditions were satisfied. To prevent R from firing, HYPGENE must make one of these conditions false. To do so, this operator posts as a new goal the negation of the predicate-calculus formula that specifies the conditions under which R fires.

Operators `Modify.Prcs.Input.To.Add.Assertion`, `Modify.Prcs.Input.To.Priv.Assertion`:

HYPGENE employs these operators when a design goal requires the modification or deletion of an object O that was created by a process R . These operators work when the existence and properties of O are dependent upon the properties of the parameter objects of R , that is, when the effects of R are a function of its inputs. These operators modify the existence or properties of O by altering the properties of the parameter objects of R . (This type of operation has been studied in planners by Chapman and Pednault [Pednault88,Chapman87].)

In order to infer what modifications (if any) will be sufficient, these operators reason about the effects of R ; in general, this is very difficult because of the procedural nature of some process effects. These operators employ an approximate solution to this problem that has fairly wide applicability in this domain. As discussed in Section 3.5.6, most chemical reactions in this domain involve object forking; thus, many processes copy their parameter objects and modify the copies in some way. If process R copies parameter object B to a new object B' , the properties of B' are largely derived from the properties of B . Therefore, if B is modified before R fires, the properties of B' will likely be modified in a similar fashion. These operators work by replacing all occurrences of B' in HYPGENE's goal stack by B . Since *all* properties of B' were not derived from B — and in fact, B' may not have been copied from any parameter object — these operators sometime fail to modify B' in the desired way.

5.5.2 Quantity-Hypothesis Design Operators

Quantitative design goals specify that the quantities of objects in a GENSIM prediction should be altered. As noted in Section 5.4, there are two types of such alterations: goals to increase

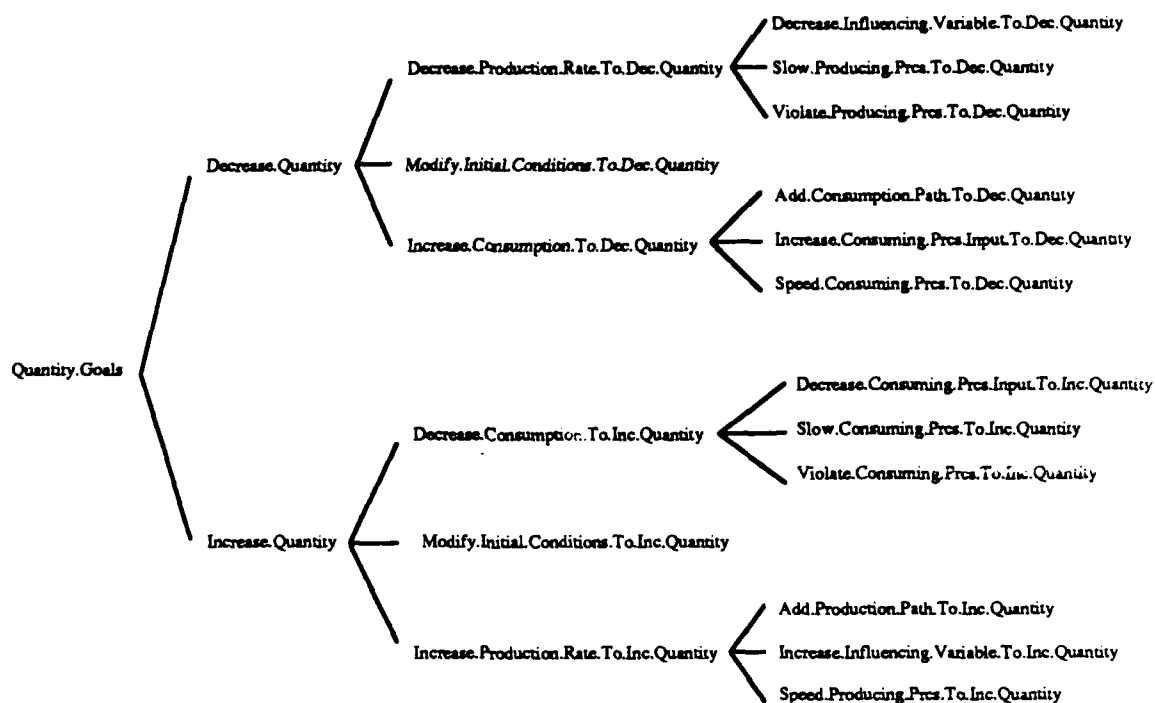


Figure 5.8: The hierarchy of quantity-hypothesis design operators. The leaves in the tree are executable operators that achieve the goals to their left.

and to decrease the amount of an object in a prediction. Figure 5.9 shows a hypothetical reaction network that will be used as an example in this discussion. Imagine that we wish to generate hypotheses about how to increase the amount of an object G that is present in the hypothetical prediction shown in Figure 5.9. In general, there are exactly three ways to increase the amount of an object:

1. Increase the amount present at the start of the experiment
2. Increase the amount produced by processes during the experiment
3. Decrease the amount consumed by processes during the experiment

Figure 5.8 shows the set of operators that HYPGENE uses to generate quantitative hypotheses. The operator `Modify.Initial.Conditions.To.Inc.Quantity` uses principle 1 to increase G by postulating that more G had been present in I_A than was originally thought.

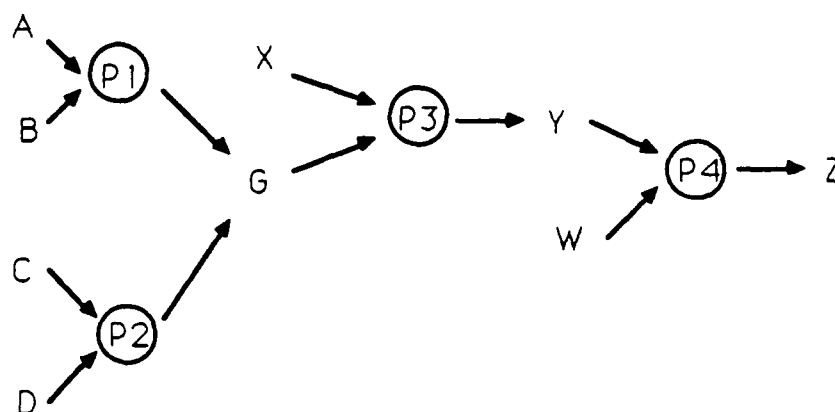


Figure 5.9: A sample reaction network. Here the process $P1$ specifies that A and B react to form G .

Principle 2 can be refined in several ways, as indicated by the members of the operator class `Increase.Production.Rate.To.Inc.Quantity` in Figure 5.8. The first is to alter the rates at which existing processes produce G . Implicit in the semantics of processes for chemical reactions is the property that increasing the concentrations of any of $\{A, B, C, D\}$ will increase the rate at which G is produced (with a restriction discussed below) — accomplished by operator `Increase.Influencing.Variable`. Second, we can think of every process as having an *intrinsic rate*, which corresponds to the rate constant for its chemical reaction. Increasing the intrinsic rate of either P_1 or P_2 will increase the production of G (operator `Speed.Producing.Prcs.To.Inc.Quantity`)⁴. The third way of increasing the amount of G produced by a process is to produce G through an additional reaction that is not currently occurring (operator `Add.Production.Path.To.Inc.Quantity`). Such a reaction may not consume any of $\{A, B, C, D\}$ — if it did then its net effect on G would be unclear.

Principle 3 can be refined in several ways, as indicated by the members of the operator class `Decrease.Consumption.To.Inc.Quantity` in Figure 5.8. These refinements are analogous to the refinements of principle 2: we can decrease the consumption of G by existing processes by

⁴What altering the rate of a process means in physical terms depends on the process in question. If the process described is an enzymatic reaction, we could alter its rate by changing the physical properties of the enzyme, such as by introducing mutations in the enzyme. HYPGENE does not possess this type of domain-specific knowledge.

decreasing the intrinsic rates of processes that consume G (P_3), or by decreasing the concentration of the objects that G combines with (X). In addition, the amount of G will be increased if P_3 no longer fires at all, which would occur if X did not exist or if a precondition of P_3 were violated. This case shares a restriction with case (2): the fact that X is a parameter object for P_3 does not necessarily imply that the chemical reaction that P_3 represents consumes the object. For example, enzymes participate in many reactions, but are not consumed by the reactions they catalyze. For HYPGENE to know whether a given object is consumed by a process, it must either be told this information explicitly, or determine it by comparing the chemical composition of the process parameter objects with the objects created by the process. In addition, this analysis cannot be local to a single process, but must be a global analysis that is applied to all objects that result from reactions involving G (in this example, Y and Z are the relevant objects). The processes that are activated by G must consume G , and none of the later processes may produce G , if we are to deduce that this network is a net consumer of G . HYPGENE does not currently perform this type of chemical analysis, but it would be fairly straightforward to implement.

In the preceding discussion we considered how to generate hypotheses to account for increased quantities of G ; accounting for a decrease in G is similar. By analogy to principle 1, we can decrease the amount of G present in the initial conditions, if G was present in I_A (operator `Modify.Initial.Conditions.To.Dec.Quantity`). By analogy to principle 2, we can decrease the production of G by decreasing any of $\{A, B, C, D\}$, or by decreasing the intrinsic rates of P_1 or P_2 , or by preventing P_1 or P_2 from firing by violating a condition on which they depend. By analogy to principle 3, we can increase the consumption of G by increasing X , by increasing the intrinsic rate of P_3 , or by firing a new process that consumes G . All objects produced by firing such a consuming process must be analyzed to determine whether they do yield a net consumption of G . They may neither produce any precursor of G (such as A), nor consume anything that reacts with G (for example, W must not be equivalent to X).

Section 7.2.3 presents hypothesis-formation problems that are solved through the use of these operators. Section 7.2.3 shows that for completeness, these operators should be used in an additional form of reasoning when formulating hypotheses as to how to alter the relative amounts of an object in two different experiments.

5.5.3 Possible Process-Design Operators

Process-design operators would alter a GENSIM prediction by modifying the theory T embodied by the process KB, and are shown in Figure 5.7. Like the initial-condition design operators, each process-design operator addresses design goals of either adding assertions to P_A , or removing assertions from P_A . The process-modification operators modify the effects, preconditions, and parameter objects of a process, so the operators are complete in the sense that they can generate any process that can be expressed using the GENSIM process-description language. Although these operators have not been implemented within HYPGENE, this section discusses how that implementation might proceed. This discussion reflects an important goal of the thesis, which is to provide a uniform and syntactically complete framework for hypothesis formation.

One important property of the GENSIM framework is that for changes to T to cause an assertion A to be removed from P_A , A must have been asserted by process firings only. The reason is that if A was present in I_A , then no modification to T can remove it from P_A , since processes cannot delete objects from a simulation. But, if A is present in P_A only because it was asserted by the firings of processes $\{R_1 \dots R_N\}$, it can be removed from P_A if we modify each process R_i such that it no longer asserts A . We can do this by either modifying R_i such that it either no longer fires in the current experiment, or by modifying R_i such that it no longer asserts A when it does fire.

Similarly, to add an assertion to P_A , HYPGENE must tailor the current experiment such that a process R fires in the current simulation and asserts A . The existence of such a process

can be achieved by any one of the following:

- Modifying the effects of a process that already fires such that it asserts A
- Modifying the preconditions of an existing process that would assert A if it fired such that it now fires
- Modifying a process that neither fires nor asserts A such that both conditions are true

Figure 5.7 shows two classes of process-modification operators: `Modify.Prcs.To.Add.Assertion` and `Modify.Prcs.To.Prevent.Assertion`. One class contains operators that can add assertions to P_A , the other contains operators that can remove assertions from P_A . We now describe each operator in detail.

**Operators `Drop.Precondition.To.Add.Assertion`,
`Generalize.Precondition.To.Add.Assertion`:**

These operators would be invoked when HYPGENE seeks to add an assertion A to P_A . They would search the PKB for a process that asserts A but does not fire in the current simulation, and make the process fire by either removing or generalizing those process preconditions that are not satisfied.

Operators `Drop.ParamObj.To.Add.Assertion`, `Generalize.ParamObj.To.Add.Assertion`:

These operators would be analogous to the operators `Drop.Precondition.To.Add.Assertion` and `Generalize.Precondition.To.Add.Assertion` except that the former operators would modify the parameter objects of the process, not the preconditions. They would search the PKB for a process that asserts A when it fires, but does not fire in the current experiment because, for some object class C listed in the parameter-object classes of the process, no instances of C exist in the SKB. One operator drops C from the process parameter-object classes, the other operator replaces C with a more general class for which some instances do exist in the SKB.

Operators Add.Effect.To.Add.Assertion, Generalize.Effect.To.Add.Assertion:

These operators would also be invoked to add an assertion A to P_A , but they would do so by modifying a process R that does not currently assert A , such that R now asserts A . They would do so by either adding A to the effects of R , or by generalizing some existing effect of R such that A is a specialization of what R asserts.

**Operators Add.Precondition.To.Prevent.Assertion,
Specialize.Precondition.To.Prevent.Assertion, Add.ParamObj.To.Prevent.Assertion,**
Specialize.ParamObj.To.Prevent.Assertion:

These operators would prevent a process R from asserting A into P_A by modifying R such that it no longer fires in the current simulation. Each operator would modify R in a slightly different way so that it no longer fires: one operator would specialize some precondition of R such that it is no longer satisfied, and another operator would specialize some parameter-object class of R to a class for which no instances exist in the SKB. A third operator would add a completely new, unsatisfied precondition to R ; a fourth operator would add a new parameter object class to R for which no instances exist in the SKB.

Operators Remove.Effect.To.Prevent.Assertion, Specialize.Effect.To.Prevent.Assertion:

HYPGENE would use these operators to prevent a process R from asserting A into P_A by modifying R such that its effects no longer assert A . One operator removes the effect that asserts A , the other specializes the effect that asserts A such that A is no longer asserted.

5.5.4 Possible Class-KB Design Operators

The initial-condition modification operators postulate changes to I_A in order to eliminate errors in P_A . Although these operators are sufficient to rectify errors in a single experiment, their

results are not generalized to apply to future experiments. For example, one hypothesis that is proposed to explain the anomalous outcome of the Jackson-Yanofsky experiment is that the leader region of the *trp* operon contains a DNA site that is a leaky transcription terminator. This hypothesis asserts that a particular part of the *trp* operon object *within this particular experiment* is a different type of object than was previously believed. The biologists soon confirmed this hypothesis, and then generalized it to the assertion that *all E. coli* *trp* operons contain a DNA site of this type. Within the GENSIM framework, this more general hypothesis would be represented as a change to the class within the CKB that defines all *trp* operons.

To accomplish this type of reasoning requires operators that generalize from modifications to I_A to create modifications to the CKB. This section briefly proposes operators that accomplish this task. These operators have not been implemented within HYPGENE. They have also not been integrated into the design framework in Figure 5.7 because it is not clear exactly where they belong. They would probably not be used to satisfy design goals resulting from a particular experiment, but would be employed after HYPGENE had successfully generated a set of hypotheses in order to generalize from those hypotheses.

Operators Create.Class, Remove.Class:

These operators would add new classes of objects to the CKB, and remove them from the CKB, respectively. I observed the use of *Create.Class* in the conceptual reconstruction in Chapter 4 when the biologists defined the class "attenuator." Note that defining a new class of objects *C* can modify a GENSIM prediction only if a process exists that includes *C* or a superclass of *C* in its parameter-object classes. Also note that simply creating a new class does not actually create a new concept; some slot in the class must be modified or the class will be identical to its parent in the taxonomic hierarchy.

Operators Add.To.Class, Remove.From.Class:

These operators would add or remove class-subclass links between existing classes in the CKB. I observed the use of Add.To.Class in the conceptual reconstruction when Ito and his colleagues postulated that the trp-synthetase protein functions as a repressor in addition to its known enzymatic function [ItoHY69] (see Section 4.4.2). In addition to postulating this additional class link, the researchers also had to create a new process that describes the activation of this repressor by a cofactor molecule (trp), and a new process that describes the binding of the activated repressor to an operator (the trp operator). Note that this is a complex hypothesis whose components are interrelated, suggesting that a macro design operator might be useful in constructing such hypotheses. Specifically, when trp-synthetase was added to the class of repressors, new processes were created to describe the behavior of trp-synthetase as a repressor. More generally, objects of a given class usually participate in a set of reactions that are characteristic of the class — repressors bind to activators and to operators.

Operators Create.Slot, Remove.Slot:

These operators would create new slots in, and delete slots from, a class.

Operators Add.Slot.Value, Remove.Slot.Value:

These operators would add a value to, and remove a value from, a class slot.

5.6 Detection of Valid Hypotheses

Whenever HYPGENE makes any change to I_A , GENSIM incrementally computes the new prediction P_A' associated with the new I_A' . HYPGENE then examines each design goal on its stack to see if it is true in the context of P_A' , that is, to see which of its design goals are satisfied. When all the goals are satisfied, HYPGENE has found a solution to its current hypothesis-formation problem. Section 6.1.5 describes this procedure in more detail. For now

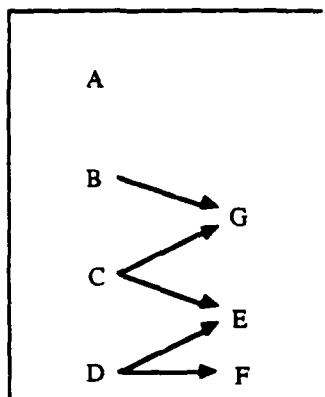


Figure 5.10: A sample reaction network.

it is important to note that a solution exists when the design goals are satisfied, but in such a solution, P_A does not necessarily match O_A exactly. More precisely, if the design goals are satisfied, it means that (referring to Figure 5.1) the assertions in AL_A have been added to P_A , and the assertions in DL_A have been removed from P_A . But, it is possible that additional assertions were added to P_A , and additional assertions were removed from P_A , such that P_A' does *not* precisely match O_A .

For example, imagine that in the prediction shown in Figure 5.10, the prediction error is that no F should exist, and some X should exist. One solution would be to remove D from I_A , thus deleting F (and E), and to postulate that some W was present, which is known to react with A to form both X and Y . This hypothesis modifies P_A to include Y and lack E , although E and Y were not mentioned in the prediction error. The reason this hypothesis is deemed acceptable is that biologists do not generally have complete, instantaneous powers of observation. They must expend time and energy to ascertain the presence of virtually every object in P_A . Thus, it is quite possible that the biologists never attempted to measure either E or Y , and simply do not know whether they are present or not.

It happens that it is easy for HYPGENE to satisfy readers who find E and Y unsettling, and who wish that P_A' did match O_A exactly. Rather than submitting $Error_A$ as HYPGENE's

design goal, we would enter a description of O_A as HYPGENE's design goal, plus universally quantified formulae that describe all the objects that were not observed. HYPGENE will then produce solutions in which P_A' does exactly match O_A . One advantage to the former approach is that the design goals are much more compact.

5.7 The Computational Complexity of HYPGENE

Although the computational complexity of the HYPGENE program is not easy to characterize, it is worthwhile to consider what factors influence the running time of HYPGENE, and how they do so. This analysis will focus on the hypothesis space that HYPGENE searches. I shall assume that the size of this space — in particular, its branching factor — is the most important influence on HYPGENE's execution time, and that the time spent processing each state is essentially constant. I shall analyze the branching factor for each type of goal that HYPGENE pursues by considering how many alternative operators HYPGENE could use to pursue the goal. Operators often produce alternative successors to the design states that they process, and the operators sometimes add new goals to these successor states, making the states harder to satisfy, so we will consider these factors as well.

Figures 5.7 and 5.8 show the operators that HYPGENE uses to achieve different types of design goals. For example, since six operators could be used to satisfy a *Decrease.Quantity* goal, there are six alternative ways that HYPGENE could try to achieve a goal of this sort. Since the current implementation of HYPGENE does not include process-design operators, they should not be counted in the branching factor for this implementation.

Here we consider the factors that influence the number of alternative successor search states produced by a given operator, and the additional goals that an operator adds to each search state. Table 5.2 summarizes the complexity of HYPGENE's operators. We shall consider one of them here in detail: the operator *Existing.Prcs.Adds.Assertion*. It attempts to add an assertion A

to P_A by finding a process R that asserts A , and by making R fire in the current experiment. The operator searches the PKB for processes that assert A , and creates a new alternative search state for each such process it finds. Thus, the branching factor for this operator is highly dependent on how many such processes exist in the PKB, which is problem dependent (this same dependency exists for every operator in the class `Process.Adds.Assertion`). When the `Existing.Prcs.Adds.Assertion` operator creates each new state, HYPGENE adds design goals to the state that express the conditions that must be true for R to fire. The cost of satisfying these additional design goals is highly dependent on the number of parameter objects for the process, on the number of objects of these types in the current simulation, and on the exact form of the preconditions of R (more precisely, on the disjunctive normal form of these preconditions). For example, imagine that R describes the reaction $A + B \rightarrow C$, and that the preconditions of this reaction are:

```
(AND (BIG B) (OR (RED A) (RED B)))
```

HYPGENE would add the following design goal to this new state:

```
(AND (OBJECT.EXISTS $A 'A)
      (OBJECT.EXISTS $B 'B)
      (AND (BIG B)
            (OR (RED A) (RED B))))
```

$\$A$ and $\$B$ are existentially quantified variables. HYPGENE will attempt to satisfy these additional design goals by trying to match $\$A$ and $\$B$ to existing objects of types A and B , respectively. There will be $N_A \times N_B$ such matches, where N_A is the number of objects of type A in the current SKB. For any given bindings of $\$A$ and $\$B$, there are two possible ways to satisfy the preconditions of the process: either $(\text{AND } (\text{BIG } B) (\text{RED } A))$ or $(\text{AND } (\text{BIG } B) (\text{RED } B))$ must be true.

5.8 Control of HYPGENE's Search

HYPGENE's operators implicitly define a large space of hypotheses. For example, since the

Operator	Branching factor
Modify.Initial.Conditions.To.Add.Assertion	1
Retract.Initial.Condition.To.Prevent.Assertion	1
Modify.Prcs.Inputs.To.Add.Assertion	process definition
Modify.Prcs.Inputs.To.Prevent.Assertion	process definition
Existing.Prcs.Adds.Assertion	PKB, SKB, process precondition complexity
Violate.Prcs.Conditions.To.Prevent.Assertion	process precondition complexity
Modify.Conditions.To.Prevent.Assertion	SKB, process precondition complexity
Modify.Effects.To.Prevent.Assertion	process effect complexity
Modify.Conditions.To.Add.Assertion	SKB, process precondition complexity
Modify.Effects.To.Add.Assertion	process effect complexity
Decrease.Influencing.Variable.To.Dec.Quantity	number of influencing variables
Slow.Producing.Prcs.To.Dec.Quantity	1
Violate.Producing.Prcs.To.Dec.Quantity	process precondition complexity
Add.Consumption.Path.To.Dec.Quantity	PKB, SKB, process precondition complexity
Increase.Consuming.Prcs.Input.To.Dec.Quantity	number of consuming process inputs
Speed.Consuming.Prcs.To.Dec.Quantity	number of consuming processes
Decrease.Consuming.Prcs.Input.To.Inc.Quantity	number of consuming process inputs
Slow.Consuming.Prcs.To.Inc.Quantity	number of consuming processes
Violate.Consuming.Prcs.To.Inc.Quantity	process precondition complexity
Add.Production.Path.To.Inc.Quantity	PKB, SKB, process precondition complexity
Increase.Influencing.Variable.To.Inc.Quantity	number of influencing variables
Speed.Producing.Prcs.To.Inc.Quantity	number of producing processes

Table 5.2: Design operator branching factors. The table lists either the exact branching factor for each operator, or the things that the branching factor depends on.

process-design operators could make arbitrary changes to any aspect of a process (its parameter objects, preconditions, and effects), they would be capable of synthesizing any process that can be expressed within the GENSIM framework. They could modify any of the normal processes in the process knowledge base, or a special null process (the latter would let the operators build an arbitrary process from scratch). Unfortunately, the use of these operators alone, in an unrestricted fashion, would produce scores of highly implausible processes. For example, consider an experiment where P_A differs from O_A in neglecting to predict the presence of a single object, B . The process-modification operators could produce many alternative modifications to T , including

1. Modifying the effects a process that already fires in E_A such that it creates B
2. Modifying a nonfiring process that already asserts B such that it fires, by removing any preconditions of the process that are violated in the current simulation
3. Modifying the null process such that it fires and asserts B , which could produce such processes as
 - (a) $I_A \rightarrow B$
 - (b) $\epsilon \rightarrow B$
 - (c) $i_1 \wedge \dots \wedge i_n \rightarrow B$ where each $i_1 \dots i_n \in I_A$

Most of these processes will be ridiculous from a biological perspective. This section discusses several methods that can be used to control HYPGENE's search of this hypothesis space so that HYPGENE is able to generate plausible hypotheses before implausible ones, and to recognize implausible hypotheses as such.

5.8.1 Simplicity

HYPGENE prefers to pursue the simplest available hypothesis, and uses a simple notion of simplicity to do so. This preference is reflected in the evaluation function that ranks items on

HYPGENE's agenda. The evaluation function ranks alternative search states on the following basis:

- States with fewer unachieved conditions on the goal stack are preferred
- The fewer the objects that have been modified by hypotheses in a given state and its ancestor states, the more the state is preferred
- Experiments have shown that one source of combinatorial explosion with HYPGENE is the code that attempts to satisfy existentially quantified conditions by binding the existentially quantified variable to existing objects in the SKB. If many objects of the required type exist, many alternative states will be created (one for each binding). For example, imagine that HYPGENE is attempting to bind the variable `$trp` to an object of type `trp`, and that five `trp` objects exist in the current experiment. HYPGENE will create five alternative search states. HYPGENE gives low preference to states that contain unsatisfied goals that were created by this type of variable binding, because my experience has shown that most such hypotheses are nonsense.

5.8.2 Domain Knowledge

Biologists possess significant amounts of domain knowledge that can be used to evaluate the plausibility of a hypothesis. For example, different chemicals are more likely to be present in some types of experiments than in others, so hypotheses that postulate that the former chemicals were present in I_A should be given high priority. Some chemicals are never present in the initial conditions of an experiment, but can be present only if a process created them. Still other objects are present only as parts of larger objects, for example, isolated genes are never present in a cell, but are always part of a larger chromosome or plasmid. Domain knowledge could also be used to choose among hypotheses formulated to explain an imaginary experiment for which GENSIM's prediction omits three objects: B_1 , B_2 , and B_3 . A host of new

processes could be formulated to account for this anomaly, including

- A single process: $[I_A \rightarrow B_1 + B_2 + B_3]$
- Two processes: $[I_A \rightarrow B_1 + B_2]$ and $[B_2 \rightarrow B_3]$
- Three processes: $[I_A \rightarrow B_1]$ and $[B_1 \rightarrow B_2]$ and $[B_2 \rightarrow B_3]$

Knowledge about what types of objects the B_n are (perhaps B_1 is a protein), and general knowledge of chemistry, may let us infer that all three objects would never be created in a single chemical reaction (this reaction might violate the law of conservation of mass — the two sides of every chemical reaction must be mass balanced), or that B_2 would never be converted to B_3 in a single step. HYPGENE does not currently use knowledge of this kind.

5.8.3 Operator Precedence

The historical reconstruction in Chapter 4 revealed that biologists tend to employ certain classes of design operators before they apply others. These preferences are not hard and fast, so they should be viewed as a tentative base for further exploration.

First, the biologists generally preferred to modify I_A before they modified T . This preference reflects the intuition that it is more plausible that a flaw exists in the single experiment at hand than that one exists in a theory developed by many people to explain many past experiments. The scientists also tended to prefer to change I_A before changing the CKB. A common type of hypothesis involved adding objects from a known class to I_A , or deleting them from I_A , or modifying the structural properties of these objects. Less common were hypotheses that created new classes of objects or modified the superclasses within which a given class of objects was defined.

I noted two common types of process modifications within the conceptual reconstruction. The first common type is the instantiation of process classes to create new process instances. The process KB records not only specific processes, but also general *classes* of processes that

scientists have observed. These process classes have been recorded in the PKB because scientists have observed a number of similar instances of these classes in the past — thus it is likely that additional processes that fall within the class will be observed in the future. For example, the process class `Repressor.Binds.Cofactor` describes the general class of reactions in which a repressor protein binds to another molecule that changes the repressor's affinity for an operator DNA-site. One way to instantiate this process is to specialize the parameter-object classes of `Repressor.Binds.Cofactor` to refer to a *particular* repressor protein (such as `Trp-ApoRepressor`) instead of the class of all repressor proteins, and to a specific cofactor (such as `Charged.tRNA.trp`) instead of the class of all molecules (this hypothesis was advanced in [ItoHY69]). This form of reasoning is somewhat similar to that developed by Greiner; he explored the idea of postulating new laws by analogy to existing specific laws [Greiner85], whereas I suggest that biologists propose new laws by instantiating existing general laws.

The second common type of process modification alters a process to depend on the presence of an activator molecule, or the absence of an inhibitor molecule. In the case of an activator, this modification involves adding a new parameter object to a process, or adding a process precondition that verifies that one object is present as a part of an existing parameter object.⁵

5.8.4 Reference Experiments

A powerful source of information for controlling the hypothesis-formation process is the existence of a *reference experiment*, E_R . A reference experiment has two important properties: Its initial conditions must be similar to those of the anomalous experiment I_A , and the predicted outcome of E_R must match the observed outcome, O_R . E_R is useful because it focuses hypothesis formation on the *difference* between I_A and I_R . This section considers three methods for using reference experiments to control HYPGENE.

⁵Neither of these modifications corresponds exactly to the use of an existing process-design operator; but because the existing operators are syntactically complete, these modifications could be accomplished by a combination of the existing operators. However, it might be more efficient to create additional macro operators to provide these common functions.

Let us first establish the relevant terminology (summarized in Figure 5.1). By definition, the outcome of a reference experiment is correctly predicted by T ; therefore

$$I_R \cup T \models P_R \qquad P_R = O_R$$

Let D_i denote the difference between the initial conditions of the anomalous and the reference experiments:

$$D_i = (I_A - I_R) + (I_R - I_A) = (I_A \cup I_R) - (I_A \cap I_R)$$

$$I_A = I_R + D_i$$

The example in Figure 5.11 illustrates some of the properties of reference experiments. Part (a) of the figure shows a hypothetical reference experiment in which GENSIM predicts that A and B react to form X , and B and C react to form Y . Since E_R is a reference experiment, GENSIM's prediction of P_R is correct. Part (b) of the figure shows an anomalous experiment in which F has been added to the initial conditions (that is, $D_i = \{+F\}$), but GENSIM's prediction does not change. However, the observed outcome of the experiment is that a new object Z is present, as well as X and Y . The reference experiment is useful because it tells us to link the appearance of Z with the introduction of F . HYPGENE might solve this problem by searching the PKB for existing processes that directly or indirectly synthesize Z from F , or by designing new processes that yield a pathway from F to Z .

We now consider three ways of using information from reference experiments to improve hypothesis formation.

Method 1: Determine Whether Hypothesis Is Consistent with E_R

The first and simplest way to employ E_R relies on the fact that, because GENSIM predicted the outcome of E_R correctly, any hypothesis H proposed by HYPGENE should not alter P_R . (A hypothesis H is a tuple $H = \{I_A', T'\}$.) This method relies on two assumptions: We assume

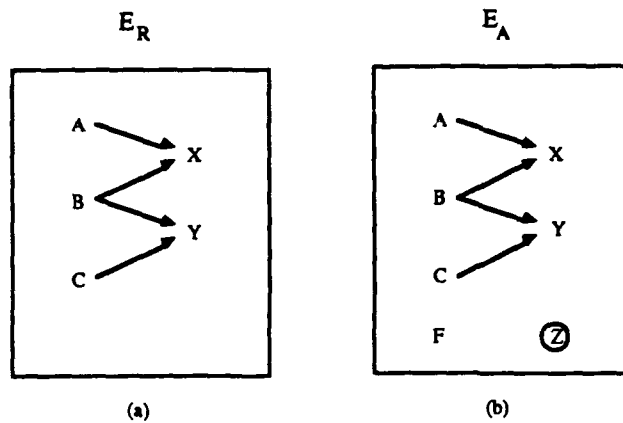


Figure 5.11: (a) A hypothetical reference experiment, and (b) a similar anomalous experiment. Introducing F in E_A causes Z to appear.

that the new theory T' should correctly predict the outcomes of both E_R and E_A . We also assume that because I_R and I_A are so similar, any unknown aspects of I_A were also present in I_R . That is, when HYPGENE proposes modifications to I_A , it postulates that the initial conditions of the experiment contained previously unknown objects, or that I_A lacked objects that we thought it contained. We assume that our description of I_R was wrong in the same way as that of I_A . If, for example, HYPGENE proposes that the Z in Figure 5.11 was produced by a reaction between F and E , where E is an unknown that HYPGENE proposes was present in I_A , we assume that E was also present in I_R .

To employ method 1, we modify I_R to yield I_R' in the same way that I_A was modified to yield I_A' , and then use the new T' to predict the outcome of the modified reference experiment, I_R' . The new prediction P_R' should match the old prediction P_R .

$$(I_R + (I_A - I_A') + (I_A' - I_A)) \cup T' \models P_R'$$

$$P_R' = P_R = O_R$$

All hypotheses that meet this criterion are compatible with both E_A and E_R ; all other hypotheses should be rejected. This criterion may be expensive to evaluate since it requires that we compute the modified prediction P_R' .

Method 2: Determine Whether Hypothesis Contains D_i

The second way to use E_R to control HYPGENE is based on the intuition that, if F caused Z to appear in Figure 5.11, then any solution to this problem must involve reactions from F to Z . Hence, we should reject any hypotheses that do not contain F . For example, F alone might create Z , or F might react with B and other objects — some of which HYPGENE might propose to add to I_A — to form Z . We might generalize this rule to form a new one: Reject hypotheses that do not contain some element of D_i . I shall show, however, that this new rule is overly general; therefore, I derive a more specific version of the rule that is correct.

Consider why the general rule is correct for the example in Figure 5.11. If F was not involved in the formation of Z , then Z must have been formed by other reactions produced by other known and unknown objects in I_A , such as A , B , and the unknown U . We assume, however, that D_i (which here contains only F) is the only difference between I_A and I_R ; therefore A , B , and U would have been present in I_R , so Z would have been observed in O_R — a contradiction.

Although the rule works for the experiment in Figure 5.11, this example is only one of several cases we must consider. In Figure 5.11, D_i is positive, meaning that F was *added* to I_R to produce I_A ; it is also possible to remove an assertion from I_R to produce I_A . In addition, the prediction error in Figure 5.11 is positive, meaning that we can obtain O_A from P_A by adding Z , rather than by removing Z . Finally, P_R and P_A make the same statement about Z — it is not predicted to be present in either outcome — although it could be present in one and absent from the other (I call the difference between P_R and P_A the *prediction difference*). Table 5.3 shows all possible combinations of the signs of the prediction error ($P_A - O_A$), the prediction difference ($P_R - P_A$), and the difference in initial conditions (D_i). The signs of the prediction error and prediction difference are not shown explicitly in the figure, but they can be derived from P_A , O_A , and P_R .

Figures 5.12 and 5.13 illustrate each of the cases in Table 5.3 with an example that shows

Assertion List	Polarity of Assertions							
	1	2	3	4	5	6	7	8
O_A	$\neg X$	X	$\neg X$	X	$\neg X$	X	$\neg X$	X
P_A	X	$\neg X$	X	$\neg X$	X	$\neg X$	X	$\neg X$
P_R	X	X	$\neg X$	$\neg X$	X	X	$\neg X$	$\neg X$
D_i	$+F$	$+F$	$+F$	$+F$	$-F$	$-F$	$-F$	$-F$

Table 5.3: Classification of reference experiments. Each column is a different type of reference experiment. For example, column 4 describes an experiment in which assertion F is present in I_A , but is not present in I_R ; GENSIM erroneously predicts that X is *not* present in P_A , when in fact X is observed in O_A ; and X was not present in P_R . Note that the value of P_A in every column is determined by the value of O_A in that column, which is why the table contains 2^3 columns instead of 2^4 columns.

a reference experiment, and both the predicted and observed outcomes of an anomalous experiment.

Our earlier argument by contradiction that D_i is responsible for $Error_A$ does not hold for cases 2, 3, 6, and 7 in Table 5.3. I call these cases the *unperturbed experiments* because when the initial conditions I_R were modified to make I_A , there was no observed change in the outcome of the experiment — modifying I_R did not perturb the outcome of the experiment. In all these cases, $P_R = O_A$ (and therefore $O_R = O_A$ since by definition, $P_R = O_R$). The unperturbed outcome suggests that D_i is not in fact responsible for the prediction error since D_i did not alter the outcome of the experiment. For example, consider case 6 in Figure 5.13. Here, X was observed in both experiments, but GENSIM predicted that X would not be created in P_A . We could explain this anomaly by proposing that the process describing this reaction is wrong in that A alone can form X — we remove F from the list of process parameter objects. There is another solution, however, *that does not involve F* : We propose that X was created by an additional pathway in *both* experiments; for example, if we know that $C + D \rightarrow X$, we propose that both C and D were present in both I_R and I_A . In this solution, F is not responsible for the prediction error — C and D are responsible. Thus, method 2 does not work

<p>$P_R = O_R$ Case 1</p> <p style="text-align: center;">$A \longrightarrow X$</p> <p>Hypothesis: F inhibits the process whereby A produces X</p>	<p>P_A</p> <p style="text-align: center;">$A \longrightarrow X$</p> <p style="text-align: center;">F</p> <hr/> <p>O_A</p> <p style="text-align: center;">A</p> <p style="text-align: center;">F</p>
<p>$P_R = O_R$ Case 2</p> <p style="text-align: center;">$A \longrightarrow X$</p> <p>Hypothesis: F does not inhibit the process whereby A produces X</p>	<p>P_A</p> <p style="text-align: center;">A</p> <p style="text-align: center;">F</p> <hr/> <p>O_A</p> <p style="text-align: center;">A X</p> <p style="text-align: center;">F</p>
<p>$P_R = O_R$ Case 3</p> <p style="text-align: center;">A</p> <p>Hypothesis: Delete the process whereby A produces X</p>	<p>P_A</p> <p style="text-align: center;">$A \begin{array}{l} \nearrow \\ \searrow \end{array} X$</p> <p style="text-align: center;">F</p> <hr/> <p>O_A</p> <p style="text-align: center;">A</p> <p style="text-align: center;">F</p>
<p>$P_R = O_R$ Case 4</p> <p style="text-align: center;">A</p> <p>Hypothesis: Create a process whereby A and F produce X</p>	<p>P_A</p> <p style="text-align: center;">A</p> <p style="text-align: center;">F</p> <hr/> <p>O_A</p> <p style="text-align: center;">A X</p> <p style="text-align: center;">F</p>

Figure 5.12: Examples of reference experiments cases 1-4. Each of the four examples shows the predicted outcome of a reference experiment on the left, and on the right shows the predicted outcome of a similar anomalous experiment, plus the observed outcome of that experiment. For each case we give a sample hypothesis that would account for the anomalous outcome. For example, in case 1 we predict that when A is present in the initial conditions of the reference experiment, I_R , that X is produced. When F is added to I_R to give I_A , we predict that X is still produced; however, only A and F are observed — X is not observed.

<p>$P_R = O_R$ Case 5</p> <p style="text-align: center;">$A \longrightarrow X$</p> <p style="text-align: center;">F</p> <p>Hypothesis: F is required in the process whereby A produces X</p>	<p>P_A</p> <p style="text-align: center;">$A \longrightarrow X$</p> <hr/> <p>O_A</p> <p style="text-align: center;">A</p>
<p>$P_R = O_R$ Case 6</p> <p style="text-align: center;"> $\begin{array}{c} A \\ F \end{array} \longrightarrow X$ </p> <p>Hypothesis: F is not needed in the process whereby A produces X</p>	<p>P_A</p> <p style="text-align: center;">A</p> <hr/> <p>O_A</p> <p style="text-align: center;">A X</p>
<p>$P_R = O_R$ Case 7</p> <p style="text-align: center;">A</p> <p style="text-align: center;">F</p> <p>Hypothesis: Delete the process whereby A produces X</p>	<p>P_A</p> <p style="text-align: center;">$A \longrightarrow X$</p> <hr/> <p>O_A</p> <p style="text-align: center;">A</p>
<p>$P_R = O_R$ Case 8</p> <p style="text-align: center;">A</p> <p style="text-align: center;">F</p> <p>Hypothesis: F inhibits a process whereby A produces X</p>	<p>P_A</p> <p style="text-align: center;">A</p> <hr/> <p>O_A</p> <p style="text-align: center;">A X</p>

Figure 5.13: Examples of reference experiments cases 5–8.

for experiments that fall within cases 2, 3, 6, or 7.

This method does work for cases 1, 4, 5, and 8 (the *perturbed experiments*, for which $O_A \neq P_R$ and $O_A \neq O_R$). Specifically, if we are considering a pair of experiments in class 4, we reject hypotheses in which the existence of X does not depend on the presence of F . For experiments in class 8, we reject hypotheses in which the existence of X does not depend on the absence of F . For experiments in class 1, we reject hypotheses in which the absence of X does not depend on the presence of F , that is, F must be an inhibitor of a reaction that leads to X . Finally, we reject hypotheses for experiments in class 5 in which the absence of X does not depend on the absence of F .

This hypothesis-evaluation criterion is a necessary but not sufficient one for correctness: even if a hypothesis H passes this criterion, it could still be rejected by method 1. Method 2, however, may be computationally less expensive than method 1 is. Method 2 has been implemented within HYPGENE.

Refining Method 2 In the remainder of this section, I show how method 2 relies on the assumption that D_i , O_A , and O_R are known with certainty. I have identified sample hypothesis-formation problems from the conceptual reconstruction in Chapter 4 in which that these entities are *not* known with certainty; however, we can augment method 2 so that it is useful without this assumption.

Jackson and Yanofsky describe a key pair of experiments in [JacksonY73]: a reference experiment in which wild-type *E. coli* is grown in excess trp, and an anomalous experiment in which an *E. coli* strain with a deletion in the trp leader region is also grown in excess trp. Although theory predicts that both strains should transcribe the trp operon at equal rates, the strain with the deletion is observed to produce trp-mRNA at a higher rate than the first strain. According to our understanding of reference experiments, the DNA deleted from the leader region must be responsible for the elevated mRNA expression. However, the biologists

formulated two hypotheses that did not contain this D_i , and that were not consistent with E_R .

The first hypothesis was that the gene-splicing procedure used to create the deletion happened to create a promoter at the deletion site, thus providing a second site for transcription initiation. The second hypothesis (the attenuation hypothesis) was that the segment of DNA that was deleted had actually contained a transcription-termination region of low strength. Both hypotheses conflict with the assumption that the exact difference between I_R and I_A is known. In the first case, the gene-splicing techniques used to create the deletion had the unforeseen effect of creating a promoter in I_A . In the second case, the gene-splicing techniques removed an object from I_R that was not thought to be present in the first place.

In general, the experimental techniques that scientists employ to make I_A differ from I_R (in this example, the gene-splicing technique that deleted the leader region) may have unpredictable effects. These effects require that we consider two additional sorts of differences between I_R and I_A : previously unknown aspects of I_R that are altered by the experimental techniques that were used to cause I_A to differ from I_R , and new unknown aspects of I_A that are introduced by the use of these techniques. Rather than assuming that all unknowns in I_A and I_R are the same, we should assume that there is a single source of differing unknowns — namely, the laboratory procedures that were used to create the difference between I_A and I_R . Biologists often have information about the types of unknown effects that their techniques can cause. For example, techniques that splice regions of DNA may also modify other nearby regions of DNA, but are unlikely to modify other chemicals in the cell arbitrarily. I shall call D_i plus these other possible differences the *extended difference*, ED_i .

Philosophers have noted that in general, there is an infinite number of differences between any two experiments, such as the time elapsed from the Big Bang and the amount of dust that has randomly settled on the laboratory bench (assuming the experiments are performed at different times). The proper approach is to consider differences in order of the likeliness of their having influenced the experiment. Here D_i should be considered before ED_i , and ED_i

should be considered before the dust on the bench.

The preceding discussion considers how to generalize method 2 to handle a real-world problem; here we consider a similar problem with method 1. The attenuation hypothesis on page 212 would be rejected by method 1 because H is not consistent with E_R . If a transcription terminator exists in the leader region, then short mRNA transcripts would be produced in both experiments because of the premature termination of transcription at that site. These transcripts were not observed in O_R . The biologists did not look for these transcripts, however, and the measurement techniques that they used were not likely to find them. (Later experiments have verified their existence [Dekel-GorodetskySE86].) To accept this hypothesis using method 1, we must generalize that method to evaluate the match between P_R' (which contains short transcripts in this example), and O_R (which does not contain short transcripts) relative to the experimental techniques that were used to observe O_R (the techniques would not have detected short transcripts).

Method 3: Reason Forward

Methods 1 and 2 augment HYPGENE's existing mode of reasoning by providing mechanisms for pruning hypotheses that HYPGENE has already generated. Method 3 is a speculative new mode of reasoning for HYPGENE. Currently, HYPGENE reasons backward from $Error_A$ to construct hypotheses that satisfy this design goal. Reference experiments, however, provide partial information about what assertions a hypothesis must contain; thus, it should be possible to reason forward from the constraints provided by reference experiments to construct a complete solution. Many of the hypotheses that HYPGENE generates by reasoning backward will fail one of the conditions imposed by method 1 or 2; HYPGENE will have wasted significant effort in generating such hypotheses. I believe that for some problems, it will be more efficient to reason forward from the information provided by a reference experiment. This form of reasoning involves incorporating information from a tester of hypotheses into a generator of

hypotheses; Dietterich and Bennett describe such reasoning in the abstract [DietterichB86].

This forward reasoning will start with information that must be present in a hypothesis (that is, a partial hypothesis), and will construct a complete hypothesis using special forward-reasoning operators. Table 5.3 provides guidance for this reasoning method, because each different case in Table 5.3 would be handled by a particular set of forward-reasoning operators. The example in Figure 5.11 falls within case 4 because the introduction of F caused Z to appear. Currently, HYPGENE reasons backward from Z to generate hypotheses, some of which will contain F . Using method 3, HYPGENE would reason forward from F either by searching for existing processes or by creating new processes that directly or indirectly yield Z . For example, HYPGENE might propose the reaction $F + A \rightarrow Z$ using an operator that designed a new process by starting with F as the first parameter object of the process, and with Z as a known product of the reaction, and that picked other parameter objects from I_A .

HYPGENE might solve the problem in case 6 of Figure 5.13 by using an operator that proposed that F is not involved in the reaction in which A yields X . The power of this operator is not obvious in this simple example; imagine that X was actually produced by a large network of reactions, one of which involved F . The operator `Drop.ParamObj.To.Add.Assertion` would generate a hypothesis for every reactant involved in the production of X , whereas the operator I propose here would focus on only F .

Related Work on Reference Experiments

Mill understood the intuition behind reference experiments when he developed his *method of difference* [Mill00]. However, Mill only considered case 4 from Table 5.3, and he did not develop methods 1, 2, or 3. In addition, Winston's strategies for using *near misses* are similar to method 2; Winston does not discuss methods 1 and 3 [Winston84].

Falkenhainer has independently recognized the value of focusing a problem solver on the difference between a positive (E_R) and a negative (E_A) example [Falkenhainer88]. His *difference-based reasoning* bears many similarities to the use of reference experiments described here. He discusses the possibilities of filtering solutions based on whether or not they contain the difference, and of reasoning forward from a difference to a solution. His approach is dissimilar from mine in that he considers *explaining* the behavior of a system (such as a bouncing ball or a car door), rather than developing a general *theory* from experiments. One reason the contrast in reasoning tasks is important is that Falkenhainer does not think in terms of *causing* a difference by applying experimental techniques, and thus does not develop the notion of the extended difference. In addition, he neither observes that solutions can be filtered based on whether they alter the interpretation of the positive example, nor analyzes this technique in enough detail to derive Table 5.3.

5.9 Summary

This chapter presented methods for solving certain types of hypothesis-formation problems. It began with a definition of the general hypothesis-formation problem, then described a restricted case of that general problem, which is the focus of this thesis. The restricted problem involves an anomalous experiment, E_A , the predicted outcome of which differs from the observed outcome. The restricted problem can include a second, reference experiment, E_R , which is similar to the anomalous experiment, but the outcome of which is correctly predicted by theory. A solution to the problem consists of a hypothesis $H = \{T', I_A', I_R'\}$ — namely, modified versions of the theory and the initial conditions of E_A and E_R , such that the predicted outcomes of both experiments are now consistent with the observed outcomes.

I treated hypothesis formation as a design problem, and described how methods developed to solve design problems can be used to formulate hypotheses. I implemented a designer of

hypotheses called HYPGENE. HYPGENE's design goal is to eliminate the error in the predicted outcome of E_A ; design goals are represented within HYPGENE in predicate calculus. There are four general classes of design goals: goals to add or remove assertions from P_A , and goals to increase or decrease the quantity of an object in P_A . HYPGENE satisfies its design goals using several types of design operators. These operators modify I_A , the PKB (not implemented), the CKB (not implemented), and generate quantitative hypotheses. Every operator addresses one of the four types of design goals. HYPGENE's overall strategy is to work backward from its design goals using operators that are likely to achieve those goals. The operators examine both the PKB and the SKB to determine what new reactions are possible, and scrutinize simulation dependency information created by GENSIM to determine what existing reactions should be altered. The chapter contained a detailed example of the reasoning that HYPGENE uses to solve a particular hypothesis-formation problem.

HYPGENE's operators are complete in the sense that they can make all possible types of changes to I_A , to the PKB, and to the CKB. Some of these operators must reason about the effects of GENSIM processes, which can be complex. I developed two approximate methods for reasoning about processes effects. One involves simulating the effects of a process on typical inputs, and observing what outputs are produced. The second involves assuming that we can achieve a desired change in the output of a process by making the same change to the input of the process, because in this domain processes often copy and modify their inputs to produce their outputs.

In designing hypotheses, HYPGENE searches a space defined by its design operators. To assess the computational complexity of HYPGENE, I considered what factors influence the branching factor of this search space. These factors include the number of operators that can address each type of design goal, and, for each operator, problem-specific factors such as the number of processes in the PKB and the complexity of particular process preconditions.

A number of methods can be used to control HYPGENE's search of this space. Some

hypotheses can be ruled out by general knowledge of biology. My analysis of the conceptual reconstruction in Chapter 4 shows that biologists exhibit preferences in their use of design operators. These preferences suggest an ordering on the use of these operators: Biologists prefer to modify I_A before they will modify T . The preferences are also manifested as common patterns of operator usage that could be bundled into macro design operators, such as creating new processes by instantiating process classes that are defined in the PKB. Another property of hypotheses that HYPGENE uses for evaluation purposes is syntactic simplicity. Finally, we discussed how to use reference experiments to filter hypotheses and to control their generation. A reference experiment, E_R , is useful because of the similarity between its initial conditions and those of E_A . We attribute some responsibility for the error in P_A to the difference between the initial conditions of the two experiments. Three reasoning methods can use knowledge from reference experiments: First, we can reject hypotheses that would alter the predicted outcome of the reference experiment, since P_R was correct. Second, we can reject hypotheses that do not contain some component of the difference between the initial conditions of the two experiments. Third, it should be possible for HYPGENE to generate hypotheses by reasoning forward from the difference between initial conditions; this type of reasoning may be more efficient in some problems than is the backward reasoning that HYPGENE now employs.

Chapter 6

Hypothesis Formation Details

This chapter addresses three loosely related topics. Section 6.1 describes the implementation of the HYPGENE program in more detail than Chapter 5 provided. Section 6.2 compares the hypothesis-methods presented in this dissertation with the methods developed by previous researchers. Section 6.3 presents several empirical lessons learned from my experiences in developing and using GENSIM and HYPGENE.

I present a detailed description of the implementation here because I believe that AI researchers have been too lax in recording the methods they use to engineer their programs. Such omission of detail can obfuscate the methods that a researcher has used to solve a problem. Some implementation details appear mundane, but if we never record mundane details, we will not have laid the foundations for more sophisticated future optimizations.

6.1 The Implementation of HYPGENE

Chapter 5 described the methods that the HYPGENE program uses to design hypotheses that eliminate errors in the predicted outcome of an experiment. HYPGENE searches a space of alternative designs. The program conducts a best-first search that is controlled by an agenda mechanism; see Figure 5.2 for an overview of HYPGENE's execution cycle. The starting search

state contains HYPGENE's design goals — a description of the prediction error. HYPGENE attempts to transform a given search state S into a solution state by executing design operators; the operators attempt to achieve unsatisfied design goals in S by modifying the initial conditions of the experiment, I_A . In so doing, an operator generates one or more new states that are children of S .

HYPGENE runs on Xerox 1100-series LISP machines (D-machines). I wrote all of the code that implements HYPGENE's search procedures, design operators, and so on, in INTERLISP. I used the KEE frame-representation system to represent the initial conditions and predicted outcomes of experiments, the GENSIM processes that describe chemical reactions, and the information contained in each search state.

6.1.1 HYPGENE's Search

HYPGENE conducts a best-first search of the design space defined by its operators. This design space is the set of possible ways in which the operators may be applied — it is the set of all partial hypotheses. The nodes in the design space are called *Dstates* (design states). Each Dstate D is represented as a KEE frame, and contains several pieces of information, including

1. A description of the unsatisfied goals at D (the goal stack).
2. Bookkeeping information, such as the name of the parent Dstate from which D was derived and the design operation that was used to create D from its parent state.
3. Complete descriptions of I_A' , T' , and P_A' . Each Dstate represents a different hypothesis that postulates different modifications to I_A and T , and hence alters the prediction P_A .

The KEEworlds ATMS (described in Section 3.5.6) is used to represent Dstates compactly [Intellicorp86]. Each Dstate is associated with a KEE world; by default it inherits all information of types 1 and 3 from its parent Dstate. We can, however, make arbitrary state-specific

changes to this information. The ATMS is used to compactly represent *both* the experiment state and the goal stack in each Dstate. I believe the latter to be a novel use of the ATMS.

HYPGENE's search is controlled by an agenda that maintains a list of the unexpanded Dstates for the current problem — the boundary of the breadth-first search. The Dstates on the agenda are ordered according to their ranking, which is computed by the evaluation function described in Section 5.8.1. HYPGENE always attempts to satisfy the Dstate that has the highest priority on the agenda.

6.1.2 Representation of Design Goals

The goal stack within each Dstate contains three separate lists of goals. Each list is an implicit conjunction of its elements. The three lists are

1. Constraints.Unresolved — Unsatisfied goal conditions that the designer has not yet attempted to satisfy (or has not yet *pursued*)
2. Constraints.Pursued — Goal conditions that the designer has pursued, but that may or may not yet be satisfied
3. Constraints.Satisfied — Those goals from Constraints.Pursued that have been achieved

The goal stack also lists the existentially and universally quantified variables in its goals, and information about the dependencies among these variables (dependencies result from the scoping of different variables in the original specification of a goal).

The elements on HYPGENE's goal stack can be in one of three forms:

1. An atomic formula or negated atomic formula containing unbound variables
2. A well-formed formula (wff) in disjunctive-normal form containing unbound variables

3. An expression with no unbound variables that is of the form (DESIGNOP param), where DESIGNOP is the name of a design operator from Figure 5.7 or 5.8, and param is a parameter to that design operator

Different design operators (*designops*) have different types of parameters; often the parameter is simply an atomic formula. A sample expression of type 3 is

(Modify.Initial.Conditions.To.Retract.Assertion
'(IS.PART 'Object.1 'Object.2))

This statement is a goal to retract the assertion that Object.1 is part of Object.2. The goal indicates *what* is to be achieved, and names an operator that should be executed to achieve it.

New goals are added to the goal stack when HYPGENE is initialized, and during the execution of some design operators. Goals are submitted to HYPGENE as arbitrary predicate-calculus formulae. Before being added to the goal stack, they are converted to either conjunctive-normal form (CNF) or disjunctive-normal form (DNF) — whichever is more compact. (This approach may or may not have efficiency advantages, but it makes the system's operation easier to understand, because the less compact form is often too complex to be readable by a human). If CNF was chosen, the conjuncts are simply appended to the Constraints.Unresolved list. If DNF was chosen, more complex reasoning is required. Ideally, each disjunct within the expression would be added to the goal stack of a *different* child Dstate of the current Dstate, because each disjunct represents an alternative way of satisfying the overall disjunctive goal. However, we cannot partition the disjuncts in this way if any two disjuncts contain the same universally quantified variable, because we would alter the logic of the goal. This partitioning is logically equivalent to rewriting a goal such as $\forall x[f(x) \vee g(x)]$ to $\forall x f(x) \vee \forall x g(x)$ — but these two expressions are not logically equivalent. Thus, the disjuncts must be partitioned into minimal-sized subsets, where there exists no universally quantified variable that is present in any two subsets. For example, we can partition $\forall x \forall y[f(x) \vee g(y) \vee h(x)]$ to alternative Dstates as $\forall x[f(x) \vee h(x)]$ and $\forall y g(y)$, and add these different subsets to different child Dstates. These

subsets can be further partitioned after the variable-binding actions described in Section 6.1.3.

This disjunct partitioning depends on the assumption that the SKB that describes experiments contains no disjunctive assertions, but only a conjunctive list of atomic formulae. If the SKB contained disjunctive assertions, we could not determine the truth of a disjunctive goal unless the disjuncts were considered together.¹

Since HYPGENE's goals are expressed in predicate calculus, but GENSIM represents experiments using frames, HYPGENE must be able to translate between the two so that, when its operators alter the predicate-calculus representation of I_A , the corresponding alteration is made to the frame representation of I_A . Therefore, each predicate used by HYPGENE has two LISP functions associated with it, called the *inversion methods*. One function alters the frame representation of I_A to *satisfy* an atomic formula that contains the predicate; the other function alters the frame representation of I_A to *violate* such a formula. The inversion methods for the predicates used by HYPGENE follow (these predicates are described in Table 3.1):

1. (OBJECT.EXISTS object class)
 - To satisfy: Add object to class
 - To violate: Delete object from class
2. (IS.PART component container)
 - To satisfy: Make component part of container
 - To violate: Remove component from container
3. (MEMB atom list)
 - To satisfy: Make atom a member of list (usually the list is a slot value, so this operation adds a value to the slot)
 - To violate: Delete atom from list

6.1.3 Expansion of Search States

HYPGENE expands a Dstate in one of several different ways, depending on the types of goal elements on the Constraints.Unresolved list of the Dstate. In the simplest case, all goals are of type

¹I thank David E. Smith for pointing out this assumption to me.

3 from Section 6.1.2 (of the form (`designop ground-formula`)). In this case, HYPGENE pursues the topmost goal G in `Constraints.Unresolved` by considering whether the design operator named in G is a leaf or a nonleaf node in the hierarchies in Figures 5.7 and 5.8. For a leaf node, the named designop is executed on its parameter. HYPGENE *refines* nonleaf Dstates by creating a new child Dstate for each descendant of the designop shown in Figures 5.7 and 5.8. A refined version of G is then pushed on the goal stack of each child Dstate. For example, we might refine the goal (`Process.Adds.Assertion (IS.PART 'Mutations.1 'Trp-Aporepressor.2)`) to (`Existing.Prcs.Adds.Assertion (IS.PART 'Mutations.1 'Trp-Aporepressor.2)`).

The other cases are more complicated because some element of the `Constraints.Unresolved` contains unbound variables — the element is of type 1 or 2 from Section 6.1.2. Elements of type 1 can have any combination of unbound universal variables (*uvars*) and unbound existential variables (*evars*); elements of type 2 must have at least unbound uvars. To process such a goal, HYPGENE first tries to locate an *independent* *ev*ar, V . A variable is independent if the goal stack does not contain any unbound variables on which V depends — variables within whose scope V was defined. (HYPGENE maintains variable scoping information). If an independent variable is found, HYPGENE attempts to use each of three methods to bind that variable:

1. Bind the variable to an existing value
2. Bind the variable to a new value that HYPGENE creates
3. Search for a process whose effects create a value to which the variable can be bound

Typically, the value of the variable will be the name of an object, and these operations correspond to binding the variable to an existing object, to creating a new object in I_A , or to finding a process that creates a object of the required type. When one of these actions is taken, a new Dstate is created and the selected value is substituted for all occurrences of the variable in question in that Dstate.

If no independent *ev*ar is found, then HYPGENE searches for an independent *u*var, and evaluates the goal-stack elements containing this *u*var under all possible bindings for the variable. For all bindings of the universally-quantified variable under which the goal elements are not currently satisfied, HYPGENE instantiates the elements and adds them to the goal stack as design goals to be achieved. If the elements contained disjunctions that could not be partitioned into separate *D*states, this binding procedure allows the disjuncts to be separated now, since they no longer contain unbound *u*vars. An additional way to satisfy violated universals is to delete every object that violates the universal (the quantification domain for each variable usually is a single class of objects).

6.1.4 Execution of Design Operators

Each *designop* is implemented as one or more LISP functions. Chapter 5 described the actions taken by the individual operators; this section describes actions that are common to all the operators. Each operator creates a record of its actions, and computes how its changes to I_A' alter P_A' .

Each operator summarizes its changes to I_A by indicating the roots (top-level object in a part-whole structure) of all objects it has modified (that is, the objects it created, deleted, or altered). The modifications to these objects can cause new processes to fire, can prevent processes that fired previously from firing, and can alter the effects of processes that fired previously and that still fire under the new I_A' .

To compute the new P_A' , HYPGENE retracts the firing of any process that acted on a modified object (as recorded by GENSIM's dependency structures), and then calls GENSIM to incrementally compute what processes are activated by the modified objects. This was a straightforward application of the simulation algorithm in Section 3.5.6, which was designed to efficiently determine which processes are activated by a newly created (or in this case, modified) object.

I tried a different approach to computing the implications of new hypotheses in an earlier version of HYPGENE. Often, changes to I_A are made with the goal of causing a process to fire or of preventing a process from firing, so it seemed natural to use this information about *why* changes are made to I_A to compute the implications of those changes. This approach is not profitable, however, because determining what process-related goal a designop is intended to achieve can be difficult, because the posting of the goal and the execution of the designop might be separated from each other by many Dstates. In addition, since it is always possible that changes to I_A will cause process firings besides those that were intended, the first type of incremental simulation would still be necessary.

Other researchers have successfully used a TMS to do much of this work for them [Simmons88]. Unfortunately, that approach would not be profitable here because my inference engine (GENSIM) was not already integrated with a TMS, and performing such an integration would not be trivial.

6.1.5 Goal Satisfaction

In most cases where HYPGENE pursues a goal element from Constraints.Unresolved, it removes the goal from that list and transfers it to the Constraints.Pursued list. The goal is not yet considered to be satisfied because there is no guarantee that the operators invoked to pursue the goal will succeed in achieving it. Instead, whenever HYPGENE finishes expanding a Dstate, it checks each of the goals in Constraints.Unresolved and Constraints.Pursued to determine whether they are satisfied. Satisfied goals are transferred to the Constraints.Satisfied list. This approach allows HYPGENE to detect the coincidental satisfaction of a goal by an action taken in pursuit of a different goal.

It is sometimes necessary to evaluate several goal clauses as a group, because one member of the group may have information about how to bind a variable contained in other clauses in the group. By recording satisfied clauses in the Constraints.Pursued list, we preserve the

binding information they contain for use in evaluating other, unsatisfied clauses.

See Section 5.6 for additional remarks on goal satisfaction.

6.1.6 Detection of Loops

HYPGENE's operators sometimes cause goal loops. For example, when pursuing a goal to make *trp* exist, HYPGENE finds many processes that create *trp*. Some of these processes, however, bind free *trp* to some other object to make a bound form of *trp*, and themselves require *trp* to fire. HYPGENE detects when a goal is repetitively posted in pursuit of itself. HYPGENE does not detect state loops — when the execution of several actions leaves the system in a state entered previously. Thus far, state loops have not been encountered.

6.1.7 Violation of Previously Satisfied Goals

Previous planning researchers have noted that for a planner to achieve several conjunctive goals, the planner sometimes must temporarily violate one previously satisfied goal in order to achieve the other goal, and then re achieve the first goal. We can view this clobbering of satisfied goals as a valuable type of planning step, since otherwise it is impossible to achieve certain goals in certain situations.

In the GENSIM ontology, however, clobbering of satisfied goals is not a useful operation because HYPGENE does not construct a temporal sequence of actions, but rather alters the set of chemical processes operating within an instant of time. Within this ontology, there is no notion of violating a goal *temporarily* and of then *reinstating* the goal, since there can be no passage of time between these two design actions. The goal is either satisfied or not for the instant of time that we are interested in.

Although this property of the ontology may seem bizarre and idiosyncratic, similar reasoning applies to processes that must be *maintained* throughout some interval of time. For example, if a plumbing problem solver had the goal of making two fluid-flow processes active

during some interval of time, but one process interfered with the other, it could not temporarily violate one flow process in order to enable the other, since duration of the flow is an essential part of the goal.

6.2 Literature Survey

This section compares my approach to hypothesis generation with approaches that other AI researchers have used to solve similar problems. Note that although most of the work discussed in this section loosely falls under the topic of scientific-theory formation, these researchers address a number of different problems.

Recall that GENSIM is given the initial conditions of an experiment plus a theory, from which it predicts the outcome of the experiment:

$$I_A \cup T \models P_A$$

The problem solved by HYPGENE is to generate modified versions of I_A and T when the predicted outcome of the experiment, P_A , differs from the observed outcome O_A . That is, given input $\{I_A, P_A, Error_A, T\}$, HYPGENE generates $\{I_A', T'\}$ such that

$$I_A' \cup T' \models O_A$$

6.2.1 Dietterich's PRE

Dietterich's dissertation addresses the problem of *theory-driven data interpretation*. The specific data interpretation task is, given: descriptions of various UNIX file-system commands such as `more` and `ls` (analogous to HYPGENE's T), plus the output obtained when a UNIX command C was executed (analogous to the observed outcome of an experiment, O_A), infer: the state of the UNIX file system (I_A) as it existed before the execution of C .

We can think of executing the UNIX command as performing an experiment on UNIX, where the output of the command is the observed outcome of the experiment.

Just as I developed a framework for representing and using theories of gene regulation (described in Chapter 3), Dietterich developed methods for representing and computing with theories of UNIX. His methods are based on the language developed for the Programmer's Apprentice system, in which programs are analogous to digital-logic circuits [RichShrobe76]. Programs consist of primitive components (corresponding to logic gates) whose input and output ports are wired together to transmit the outputs of one component to the inputs of others. A primitive element is a LISP procedure such as CONS. New primitives can be constructed by treating a circuit as a black box with an assigned name and input-output ports.

A given data-interpretation problem consists of the program circuit describing a UNIX command, plus observed values for the output ports of the circuit. PRE infers the inputs to the command (the state of the UNIX file system) by reasoning backward through the logic circuit for the command from the values on its output wires. Dietterich uses constraint-propagation techniques to infer the inputs to primitive elements within the circuit from their outputs, until the inputs of the command itself are derived.

The following similarities and differences exist between our approaches. First, the characteristics of the two domains necessitate different modeling techniques. Although the program circuit for a UNIX command corresponds closely to the dependency structures produced by GENSIM simulations, the command descriptions (and their associated circuit-like definitions) are an *input* to PRE, whereas GENSIM *computes* the dependency structure from the initial conditions of the experiment plus the process library. In the UNIX domain, an experiment consists of executing a single command in the context of the file system. In our domain, the initial conditions of the experiment in effect determine what "commands" (processes) are executed (as do the objects created by the execution of processes). In addition, the UNIX domain does not require any qualitative reasoning about numerical state values, so PRE does not address this issue.

Another difference between PRE and HYPGENE is that PRE computes I_A from $\{O_A, T\}$ whereas HYPGENE computes $\{I_A', T'\}$ from $\{I_A, P_A, Error_A, T\}$. Thus, PRE is concerned with interpreting a correct prediction whereas HYPGENE seeks to debug a faulty prediction. (PRE is sometimes given partial information about the initial state of UNIX.) Although HYPGENE could compute $\{I_A', T'\}$ from $\{P_A, T\}$ alone, I_A' is usually sufficiently similar to I_A that it is easier for HYPGENE to compute I_A' by modifying I_A than to compute it from naught. (In general, many scientific experiments do contain significant information about their starting conditions.) Also, PRE has no framework for modifying its theory T ; HYPGENE does have such a framework.

PRE and HYPGENE both reason backward through a dependency network, but the two programs use slightly different goals and power. PRE reasons backward from O_A because it is not able to compute a prediction — it has at most a partial description of I_A . HYPGENE reasons backward from the *difference* between P_A and O_A . PRE propagates the known output of a circuit gate back through that gate to deduce what input(s) to the gate could have caused the output. HYPGENE uses its operators to work backward from an unpredicted or incorrectly predicted assertion to postulate experimental conditions or theory modifications that would account for this assertion. Whereas PRE examines a fixed set of circuit components, HYPGENE's design framework essentially gives HYPGENE the power to modify the inputs to existing "components", to "wire" known types of components into the biochemical-reaction "circuit" with appropriate inputs (by postulating that additional processes fire), or to create new types of components (by postulating the existence of new objects) and to wire them into the reaction network. Finally, PRE's constraint-propagation approach is a deductive framework that cannot be extended easily to the task of modifying PRE's theory T . A modification of T would alter the set of constraints in the statement of the problem; thus it is not a use of constraint propagation per se.

The techniques that Dietterich developed for regressing goals through Programmer's Assistant-style programs could be valuable for HYPGENE. The latter's design operators must perform a similar analysis on domain processes to determine what initial conditions must be established to achieve required effects through the execution of a process. Dietterich's analysis of recursive procedures is particularly difficult and useful.

6.2.2 Simmons' GORDIUS

Simmons' recent work is probably the most similar of any to this dissertation. His dissertation addresses geologic *interpretation* problems, which involve a geologic region that has changed over time due to the actions of geologic processes. Given descriptions of the region before and after the change, his system proposes a sequence of geologic processes that could have transformed the starting state to the ending state (such as faulting and erosion). Simmons calls a proposed sequence of processes an *interpretation*. He proposes a strategy called GTD — generate, test, and debug — to solve this problem. GTD uses a rule-based system to generate an approximate interpretation, and then tests this solution with a simulator. If the interpretation is incorrect, a debugger attempts to eliminate its flaws.

More precisely, the solutions GORDIUS seeks are sequences of instantiated processes of known types, such as an uplift of 10 meters followed by erosion of 5 meters followed by a tilt of 20 degrees. His simulation system simulates the effects of these processes on the starting geologic state using techniques similar to those employed by GENSIM. Among the differences between GORDIUS and GENSIM are that GORDIUS represents the passage of time and reasons about quantitative system state variables. In addition, the preconditions of a GORDIUS process are necessary but not sufficient conditions for execution of the process — GORDIUS treats geologic processes as "acts of God" (they are imposed by forces that GORDIUS does not model), whereas GENSIM processes must occur if their preconditions are satisfied.

Just as HYPGENE is invoked when GENSIM's prediction does not match an observation,

the GORDIUS debugger is invoked when the final geologic state predicted by the GORDIUS simulator does not match the known final state of the region. The debugger considers each bug in the prediction (for example, the height of a region might be incorrect) and examines dependency structures that were produced by the simulator to determine what assumptions the bug depended on (such as the amount of erosion that occurred at some time). GORDIUS considers assumptions about

- The events (processes) that occur
- The bindings of parameters (such as the height attained by an uplift process)
- The relative ordering of events
- The persistence of attribute values
- The persistence of objects
- The existence of objects of a given type

These assumption types are similar to the operators implemented for HYPGENE. For example, event-occurrence hypotheses would be generated by the operators `Existing.Prcs.Adds.Assertion` and `Violate.Prcs.Condition.To.Prevent.Assertion` (see Section 5.5.1). Simmons does not propose assumptions that correspond to HYPGENE's process-modification operators, although he says that adding new classes of assumptions to GORDIUS is easy to do.

One disadvantage to the use of assumptions rather than operators is that the GORDIUS simulator explicitly inserts all possible types of assumptions at every possible point in the ATMS dependency structure. For example, a persistence assumption is associated with every object value at every point in time. In large simulations, *many* assumptions are generated, at a significant cost. HYPGENE's operators let it represent these assumptions implicitly. For example, the `Violate.Prcs.Condition.To.Prevent.Assertion` operator causes HYPGENE to attempt

to retract an assertion A from P_A by violating some precondition of every process that asserted A . GORDIUS would have to record each satisfied precondition explicitly as an assumption.

Simmons' approach shows the advantages of having a simulation system that is closely coupled to a TMS; special programming was required to create the dependency structures that HYPGENE uses.

Simmons' regression of values through process descriptions is more sophisticated than that used by HYPGENE. Simmons uses a constraint language to indicate what process input values will cause a process to yield a desired output value. Regression is possible because the effects of many GORDIUS processes can be described as mathematical functions that are relatively easy to invert. Like GENSIM processes, the effects of some GORDIUS processes are sufficiently complex that no regression can be done. Simmons does not employ the heuristics described in Section 5.5.1.

6.2.3 Rajamoney's COAST

Rajamoney has addressed the problem of theory revision in the domain of naive physics. His system formulates hypotheses to revise a theory when its predictions do not match observation, designs new experiments to discriminate between hypotheses, and evaluates competing hypotheses based on past experiments and on aesthetic criteria.

Rajamoney's approach to hypothesis formation focuses exclusively on revising his domain theory T (which is expressed in qualitative process theory) when the predicted rate of change of some quantity (such as the rate of evaporation of a fluid) does not match the observed rate. Theory revisions are performed by operators that are strikingly similar to HYPGENE's process-modification operators. If, for example, his domain theory predicted that a quantity decreases when the quantity was observed to stay constant, his system would locate the process responsible for predicting the decrease, and would postulate that the process is inactive (corresponding to HYPGENE's *Modify.Condition.To.Prevent.Assertion* operator class (see

Section 5.5.3)). Specific ways of rendering the process inactive include creating new process preconditions and narrowing the scope of existing preconditions — corresponding to HYPGENE's operators `Add.Precondition.To.Priv.Assertion` and `Specialize.Precondition.To.Priv.Assertion`. Rajamoney does not define operators for modifying the `Individuals` field of a process (the equivalent of GENSIM's `Parameter.Object.Classes`), so apparently his system is not syntactically complete.

One important difference between COAST and HYPGENE is that COAST is able to generate abstract hypotheses, to test them for correctness, and then to refine those abstract hypotheses that are not rejected. The advantage of this approach is that, if an abstract hypothesis is rejected, all specializations of that hypothesis are also ruled out in one fell swoop. Although HYPGENE does not include this capability at present, such reasoning should be easy to incorporate into HYPGENE's framework: HYPGENE already generates abstract hypotheses such as `Process.Adds.Assertion`, but it does not test them before refining them.

Rajamoney does not define operators that modify either the initial conditions of the experiment, or the CKB. In addition, COAST cannot formulate hypotheses to account for prediction errors in which extra objects are present in a prediction, or in which observed objects are missing from a prediction. Although Rajamoney says little about the implementation of his system, it does not appear to include a sophisticated planner. This lack would prevent COAST from reasoning about processes with complex preconditions, or about complex networks of processes — most of Rajamoney's sample hypothesis-formation problems involve predictions in which only one process executes.

Rajamoney's use of previously stored experiments to filter hypotheses is the same as my first method for employing information from reference experiments to filter hypotheses (see Section 5.8.4). Rajamoney does not use either method 2 or method 3 from Section 5.8.4. Rajamoney does address the problem of designing new experiments to test hypotheses, whereas I do not.

Rajamoney also addresses the issue of evaluating competing theories using "aesthetic criteria," after the preceding empirical criteria have been exhausted. He proposes aesthetic-evaluation functions that are based on such properties as syntactic features of the processes contained in a theory, the syntactic complexity of explanations generated by the theory, and the number of predictions made by the theory. I have not addressed aesthetic evaluation of theories. Unfortunately, Rajamoney's criteria are defined too imprecisely to evaluate in the context of my historical study of attenuation.

6.2.4 Wilkins' ODYSSEUS

Wilkins' dissertation describes methods for apprenticeship learning, developed in the domain of medical diagnosis [Wilkins88]. His apprenticeship-learning program improves the diagnostic abilities of the expert system NEOMYCIN by watching how an expert doctor solves diagnostic problems. ODYSSEUS learns from the questions a doctor asks during a consultation by assuming that, if the diagnostic knowledge contained in NEOMYCIN cannot be used to provide some justification for a question asked by the doctor, there is a deficiency in NEOMYCIN's knowledge. ODYSSEUS both suggests alternative changes to NEOMYCIN's KB that would remedy this deficiency, and evaluates these alternative KB repairs.

For example, imagine that the doctor asks her patient whether he has a headache, and the NEOMYCIN KB provides no justification for this question in the current problem-solving context. ODYSSEUS might postulate that the condition of having a headache is evidence for a disease that the doctor is considering, which would explain why the question was asked.

The problem and methods described by Wilkins bear a number of similarities to those presented in this dissertation. The doctor's questions are observations to be explained (O_A). ODYSSEUS tries to explain the questions using NEOMYCIN's knowledge of medicine, plus knowledge of the current problem-solving goals (such as knowledge of what diseases are considered likely at a given moment); this knowledge constitutes I_A . ODYSSEUS uses NEOMYCIN's

strategic knowledge of diagnosis (encoded as a set of metarules) to link I_A to O_A . A metarule might say that, if condition X is evidence for disease Y , and NEOMYCIN is trying to establish whether Y is present, then ask whether X is present. For every question Q asked, ODYSSEUS attempts to find some path from I_A to Q using the metarules in T .

If ODYSSEUS cannot find such a path, there exists no chain of metarules with satisfied left-hand sides derives Q from I_A . The left hand sides of these metarules are conjunctive lists of atomic formulae. To generate KB repairs, ODYSSEUS relaxes the requirement that, for a metarule to fire, every conjunct in the metarule must be satisfied. It tries to generate explanations containing rule firings in which a single conjunct within a rule precondition is not satisfied. Unsatisfied conjuncts indicate propositions that should be added to the NEOMYCIN KB. Each such proposition is an alternative KB repair. This procedure is similar to that used by HYPGENE's operator Existing.Prcs.Adds.Assertion, which attempts to satisfy the preconditions of a process in the PKB such that the process fires and adds an assertion to P_A . An important difference is that GENSIM process preconditions contain conjunctive lists of arbitrary predicate-calculus formulae — not atomic formulae — so HYPGENE must perform more complex reasoning to determine how to satisfy a given set of preconditions (see Section 6.1.3). Another important difference is that ODYSSEUS does not provide any facility for modifying the NEOMYCIN metarules — NEOMYCIN's theory of diagnostic strategy is assumed to be correct, whereas the process-design operators described here are able to modify GENSIM's theory.

Wilkins also addresses the problem of evaluating the alternative repairs that ODYSSEUS proposes to NEOMYCIN. His approach is to use a *confirmation theory* that can evaluate the goodness of any atomic formula that ODYSSEUS proposes to add to NEOMYCIN's KB, based on the predicate in that formula. This technique is valuable and could be employed by HYPGENE. One important point is that Wilkins assumes that the confirmation theory is devoid of flaws — he makes no provision for improving the confirmation theory.

6.2.5 Lenat's AM and EURISKO

Lenat has constructed two well-known programs for automated discovery: AM and EURISKO. They differ from HYPGENE in two important respects. First, AM addresses a different problem than HYPGENE does, because AM is not concerned with the empirical relationship between the predicted and observed outcomes of an experiment. Second, both AM and EURISKO are designed for domains that do not have an invertible performance standard (which is described later in this section).

AM's task is to discover interesting concepts in mathematics. AM seeks these concepts by performing a heuristic search in which it uses one set of heuristics to generate new mathematical concepts from old ones, and then evaluates the "interestingness" of the new concepts using a second set of heuristics. A search for interesting concepts is different from work in the empirical sciences in which scientists are concerned with developing a theory that correctly predicts outcomes of experiments. Workers in the natural sciences receive little or no credit for developing theories that are only interesting; their theories must be grounded in empirical data to have any serious value. The theories that AM produces have no particular relationship to any observable phenomenon. HYPGENE is concerned with exploring just this relationship.

HYPGENE's general framework of design may, however, be applicable to AM's problem. In this approach, HYPGENE's goal would be to design interesting mathematical concepts from less interesting ones. To do so, we would view interestingness as AM's performance standard, where interestingness is computed from a concept using Lenat's interestingness heuristics, much as a prediction is computed for an experiment using a theory:

$$\text{evaluation_heuristics}(\text{concept}) \Rightarrow \text{interestingness_value}$$

Since the goal is to maximize interestingness, HYPGENE would have to reason backward through the dependency structure of an interestingness computation to determine how to modify its concepts to increase their interestingness. Unfortunately, it is not clear how to invert the interestingness heuristics, that is, how to alter a given concept such that we increase

the interestingness value that a heuristic would assign to it. In contrast, GENSIM's reasoning is much easier to invert.²

EURISKO is based on many of the same ideas as is AM, but it was applied to many domains and was able to learn new evaluation heuristics. Some of these other domains possessed a much clearer performance standard than AM did. For example, EURISKO was used to design space fleets that engage in battles that are either won or lost. EURISKO reasons about the success or failure of a given fleet to create new heuristics that suggest how to modify the fleet for better performance. Unfortunately, Lenat does not describe how EURISKO performed this task, so we cannot compare EURISKO's approach to that used by HYPGENE. In any event, HYPGENE does not create new evaluation heuristics.

6.2.6 The Early Chemists

Researchers at Carnegie Mellon University and the University of California at Irvine have explored the discovery process with five different computer programs, all of which have reproduced discoveries from early chemistry. These programs are called BACON, GLAUBER, STAHL, STAHLp, and DALTON; they are described in [Langley87,RoseL86]. BACON finds quantitative equations (laws) consistent with quantitative measurements of a physical system: GLAUBER finds qualitative chemical-reaction laws consistent with a set of observed chemical reactions.³ STAHL, STAHLp, and DALTON postulate container-component relationships among the chemicals in a reaction.

These programs differ from HYPGENE in terms of both the scientific reasoning tasks they address, and the methods they use to perform these tasks. The chemistry programs use purely data-driven reasoning to solve problems: their inputs are data from experiments, and their

²Note that, by analogy with HYPGENE's modifications to a theory, a second possibility would be to increase the interestingness value by altering the interestingness heuristics such that they gave a more favorable evaluation of the current concepts! EURISKO learns new evaluation heuristics.

³Note that Langley et al. use the word *qualitative* in a different sense than I used it in Chapter 3: their qualitative laws are production rules that include no representation of quantities.

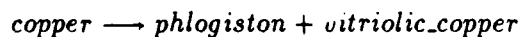
outputs are theories that explain the data. Unlike HYPGENE, they do not begin with a theory and use it to predict experimental outcomes. Thus, their methods apply to a scientific field in its infancy, when no theory exists to predict experimental outcomes.

The chemistry systems can consider data from many experiments simultaneously, whereas HYPGENE considers at most two experiments at once. For example, STAHLp computes hypotheses consistent with a large number of simple experiments (an experiment might involve a single reaction), whereas HYPGENE computes hypotheses consistent with a small number of complex experiments. Their methods are similar in that both manipulate reasoning dependency-structures. HYPGENE reasons about dependencies between reactions in a single GENSIM prediction to infer how that prediction might be altered. In contrast, STAHLp uses techniques similar to constraint propagation to infer component structures from descriptions of several experiments. These experiments are sometimes inconsistent; when the constraint propagator detects an inconsistency, STAHLp uses a dependency structure created during the constraint-propagation process to propose changes to the experiments that will render them consistent. Thus, HYPGENE reasons about the correctness of a single experiment, and STAHLp reasons about the consistency of multiple experiments. Because HYPGENE's processes (reactions) are so complex, it must manipulate complex predicate-calculus formulae when it traverses this dependency structure. Because STAHLp processes are very simple, it need only reason about what objects were or were not present in each experiment.

Both this dissertation and the work of Langley and his colleagues view discovery as a search process, and employ operators to search a space of solutions. But a number of important differences exist between the two approaches:

- HYPGENE's input describes a *difference* between prediction and observation, and a starting theory. HYPGENE does not derive a theory from naught. The chemistry programs take only experimental data as input.

- HYPGENE works *backward* from the difference to formulate hypotheses, and is thus guided by the dependency structure created by GENSIM. The chemistry programs other than STAHLp apply their law and component-structure generation operators in a forward search that has a small amount of heuristic guidance. For example, when the DALTON program searches for the component structure of water, it blindly generates molecular formulae such as HO , H_2O , HO_2 . This approach would take quite some time to find the formula of the compound glucose: $C_6H_{12}O_6$.
- HYPGENE uses the difference between the initial conditions of an anomalous experiment and a reference experiment to focus its search. Since the chemistry programs do not predict the outcomes of the experiments they consider, they have no notion of a correctly predicted versus an incorrectly predicted outcome, and so they cannot use this information.
- Unlike BACON, HYPGENE does not have operators for generating algebraic mathematical laws. But HYPGENE can generate hypotheses about how to modify the quantities of objects produced in an experiment.
- The expressiveness of the language used by the chemistry programs for representing qualitative laws (chemical reactions) is extremely limited. For example, they cannot represent reactions that are predicated on any type of logical condition, such as the states of the reactants. All of their processes can be expressed in a single line such as



whereas HYPGENE process definitions are sometimes a page in length. The need to manipulate complex process definitions required that HYPGENE possess the ability to manipulate complex predicate-calculus formulae.

- Langley and his colleagues have implemented the operators that generate new reaction laws within the chemistry programs. I have not implemented the process-design operators within HYPGENE.
- The operators used by the chemistry programs to generate qualitative laws and component models are relatively domain-specific, for example, they explicitly refer to the predicate *component-of*. The HYPGENE operators do not mention domain-level concepts, but manipulate processes and predicate-calculus assertions. Thus, HYPGENE's operators are more general.
- The chemistry programs generally consider single chemical reactions, whether they generate laws to describe these reactions or component models for the objects in those reactions. HYPGENE considers complex networks of reactions.

6.2.7 DENDRAL

The DENDRAL program solves the problem of elucidating the chemical structure of an organic compound C given the chemical formula of C and a mass spectrum of C [Lindsay80]. That is, given O_A (a mass spectrum), plus a partial description of I_A (the chemical formula of C), plus a partial theory of mass spectroscopy T , DENDRAL computes a full description of I_A (the chemical structure of C).

DENDRAL uses a plan, generate, and test method to solve this problem. The planning phase deduces constraints on the structure of the compound from the mass spectrum. These constraints specify chemical groups that the structure must or must not contain. A generator of molecular structures that computes all chemical structures that are consistent with both the chemical formula of C and the constraints deduced in the planning phase. In the test phase, DENDRAL uses its partial theory of spectroscopy to simulate the passage of each chemical structure produced by the generator through a mass spectrograph. The simulated spectrum

is compared against the spectrum observed for C . The proposed structure whose simulated spectrum has the closest match to the observed spectrum is deemed the best structure. The observed mass spectrum may contain noise.

One important characteristic of DENDRAL is that the chemical formula of C completely determines a finite (but often large) space of chemical structures for C . HYPGENE, on the other hand, always faces an infinite space of possibilities for I_A' and T' since arbitrary new objects with arbitrary properties can always be added to I_A . Also, in DENDRAL's domain it is possible to design a generator that incorporates solution constraints from a planner. The experimental constraints from the molecular biology domain are so diverse that it is not clear how to incorporate them in a hypothesis generator.

Conversely, it would be difficult to use HYPGENE's approach to solve DENDRAL's problem because the programs address different types of problems. HYPGENE starts with full descriptions of $\{I_A, T, P_A\}$ and the dependency structure for the simulation. It then works backward from the difference between P_A and O_A to hypothesize changes to I_A and T that would account for this difference. If DENDRAL used this approach, it would have to start with some description of the chemical structure of C (I_A), when in fact it starts with only the chemical formula of C . It would then use a complete theory of mass spectroscopy to predict the spectrum (P_A) observed for C , when in fact it has only a partial theory of mass spectroscopy. Then it would compare the predicted spectrum to the observed spectrum, and work backward from this difference to postulate changes to the initial chemical structure or to the theory of spectroscopy, to account for the prediction error. The approach used by HYPGENE requires that we reason backward through the dependency structure of DENDRAL's prediction — that the chemical cleavage rules that embody DENDRAL's theory of spectroscopy are invertible. (These cleavage rules are the equivalent of processes in DENDRAL's domain.) Since a cleavage rule specifies where a large fragment breaks to form two smaller fragments, the inverse of such

a rule would work backward from combinations of fragments in the mass spectrum to postulate larger molecules that broke apart to form the observed fragments. Unfortunately, there would be *many* such combinations of fragments, which would yield a large space of alternative hypotheses.

6.2.8 Meta-DENDRAL

The Meta-DENDRAL program learns cleavage rules that describe what molecular bonds will break when chemical compounds of a certain class pass through a mass spectrograph [BuchananM78]. The DENDRAL program uses this fragmentation theory in its task of elucidating chemical structures from a mass spectrum. DENDRAL finds I_A (a chemical structure) given T (a fragmentation theory) and O_A (a mass spectrum), whereas Meta-DENDRAL learns T given I_A and O_A .

Meta-DENDRAL learns a fragmentation theory in three steps. The INTSUM program uses a half-order theory to learn overly specific rules that describe the breakage of a single bond in an entire molecule. Then the RULEGEN program generalizes the rules produced by INTSUM. Next, the RULEMOD program weeds out duplicate rules from those produced by RULEGEN, and eliminates rules that produce false-negative predictions with respect to the mass spectrum.

Thus, Meta-DENDRAL essentially uses a generate-test-debug approach — the INTSUM program generates many rules and keeps only those that make true-positive predictions; the RULEGEN and RULEMOD programs debug the rules by generalizing them and eliminating certain rules. Meta-DENDRAL does not work backward from a prediction as HYPGENE does; my methods probably would not work well in this domain for the same reasons they would not work well in DENDRAL's domain (see Section 6.2.7). An interesting similarity between Meta-DENDRAL and my approach is that the INTSUM program uses a half-order theory of spectroscopy as its initial rule generator. In a sense, INTSUM computes its output by modifying this half-order theory. HYPGENE's process-design operators would also modify the theory

HYPGENE starts with, but generally HYPGENE would make much less drastic changes to a theory than does INTSUM. HYPGENE would make those changes in a much more goal-directed fashion, by considering how a modification affects the overall GENSIM simulation dependency trace, rather than by considering whether each rule in isolation yields true or false predictions.

6.2.9 Koton's GENEX

Koton constructed a system called GENEX that solves hypothesis-formation problems in regulatory genetics [Koton85]. She implemented three different versions of GENEX, each of which uses different problem-solving methods. Given a partial description of an operon, and a description of the behavior of the operon, GENEX proposes a detailed description of the operon, and (for Versions 2 and 3) an explanation of why the proposed structure of the operon would cause the observed behavior. For example, if GENEX was told that a repressor protein contained a mutation, it might propose that the behavior of the operon would be explained if the mutation existed within the operator-binding site of the repressor.

Version 1 was essentially an expert system that used heuristic classification to assign the given operon to one of several predefined regulatory classes. Version 2 modeled the structure and behavior of genetic objects and processes. The key idea in this version was that partial descriptions of objects introduce uncertainty in object states; for example, the statement that a repressor contains a mutation implies that different parts of the repressor might be in functional or nonfunctional states. GENEX generated hypotheses by simulating all possible functionalities of these parts. When the simulated behavior of the operon under a given assumption about a part's functionality matched the observed behavior, GENEX proposed that the part had that functionality. Koton found that this approach was computationally expensive because the number of simulations was exponential in the number of uncertainties. Version 3 combined her ideas from Versions 1 and 2; first the operon is assigned to a predefined class of regulatory mutants based on its behavior, then GENEX uses the regulatory class to filter the set of part

functionalities that it simulates.

GENEX device models are similar to those of GENSIM, but are considerably less complex. GENEX is able to formulate fewer classes of regulatory hypotheses than is HYPGENE — Versions 1 and 3 require that the user predefine a set of regulatory classes, and Versions 2 and 3 can formulate only hypotheses that involve mutations. GENEX does not provide a framework for modifying its theory of genetics.

6.2.10 Kulkarni's KEKADA

Kulkarni addresses a number of issues involved in scientific discovery [Kulkarni88]. Although his paper is entitled "The Strategy of Experimentation," it discusses several issues that are given equal footing: how to generate and choose scientific problems for study, how to propose experiments, and how to generate and rank hypotheses. Kulkarni's overall approach is similar to that taken here; he has written a program that reproduces a historical instance of discovery: the discovery by Krebs of the ornithine cycle, which organisms use to synthesize the chemical urea.

Unfortunately, Kulkarni's description of his methods lacks detail. KEKADA is an agenda-based system that employs a number of heuristics to accomplish the tasks we have listed. Kulkarni provides brief English descriptions of these heuristics that appear reasonable, but he does not give sufficient detail that we can determine how we might apply the heuristics to HYPGENE's task. For example, it is difficult to compare KEKADA's hypothesis generation to that of HYPGENE because Kulkarni does not describe either the representations of KEKADA's objects and theories, or its hypothesis-generation operators. It is also not clear whether a pure generate-and-test approach is used, or whether hypothesis generation is goal-directed.

6.2.11 Tong's DONTE

Tong has studied the design process in intricate detail [Tong88]. His main contribution to design is a technique called goal-directed planning (GDP). GDP plans the design process by searching a space of design plans for the plan that best suits the design problem at hand. The plan includes information about the ordering of design operations, and about decisions that select among alternative design operations. GDP can also reformulate the design space to decrease interactions among design operators. It performs *rough designs* at a gross level of abstraction to produce a lower bound on the costs of different designs.

Tong's approach to design is considerably more sophisticated than mine is. HYPGENE's efficiency probably would be improved if HYPGENE employed GDP. Conversely, the hypothesis-design task probably would be an interesting test of Tong's ideas.

6.2.12 Other Work

Here I briefly mention additional past research that is related to my work in hypothesis formation.

Hayes-Roth has formulated a number of heuristics for rectifying refuted theories; his heuristics are quite similar to the process design operators I described in Section 5.5.3 [HayesRoth82].

Davis's TEIRESIAS aids a user in debugging a diagnosis computed by MYCIN [Davis76]. TEIRESIAS works backward through MYCIN's rule stack to determine why a MYCIN solution was wrong. TEIRESIAS relies on advice from the user more strongly than does HYPGENE.

Soo addressed the problem of proposing enzyme-reaction mechanisms given experimental data about the rate at which an enzyme reaction occurs under varying concentrations of the inputs, outputs, and inhibitors of that reaction [Soo87]. Soo solves this problem using the generate-and-test approach. He has deduced that every enzyme is described by one of a fixed number of preenumerated reaction mechanisms. For each member of this set of reaction mechanisms, he generates predictions of the enzyme's behavior under all possible qualitatively

different experimental conditions. These predictions are matched against abstract descriptions of the experimental data to determine which reaction mechanisms are consistent with the data.

Much past work on model-based diagnosis is relevant to the problem of hypothesis formation [Genesereth84,Davis84,deKleerW86]; however, two important differences exist between device diagnosis and scientific-hypothesis formation. First, a diagnostician usually assumes that her theory of how a device operates is correct, but that her description of the device is in error. For example, she assumes that she has a valid theory of how AND gates function, but that a particular gate within a circuit that she thought was an AND gate is in fact broken, and is thus not truly an AND gate. Thus, diagnosticians usually modify I_A , but not T .

Second, at least in digital-circuit diagnosis, we make a restricted set of assumptions about what changes to I_A are possible — generally, we assume that wires are broken or that gates are not functioning as expected. In molecular biology, on the other hand, scientists postulate the presence of arbitrary additional objects in an experiment; these objects can cause many additional reactions. If we draw an analogy between chemical objects and gates, and between reactions and wires, these biological hypotheses correspond to diagnostic hypotheses that a circuit contains arbitrary additional gates, connecting wires, input ports, and output ports. Such diagnostic hypotheses are quite rare, although the bridge-fault hypotheses proposed by Davis's system form a restricted class of hypotheses of this sort [Davis84].

6.3 Lessons Learned from GENSIM and HYPGENE

The remainder of the chapter presents lessons learned from experiments with GENSIM and HYPGENE that should make construction of similar programs easier in the future. These lessons address the subjects of how to write process preconditions that can be easily transferred to other processes to facilitate the creation of new processes, and of what types of preconditions can be troublesome during the inversion phase of HYPGENE's reasoning. There is apparently

a conflict between the applicability of preconditions to new processes, and the efficiency with which they are inverted: general preconditions are inefficient to invert. I also discuss a conflict, which arose during the implementation of HYPGENE, regarding whether the SKB should be represented using frames or predicate calculus.

6.3.1 On Writing Process Preconditions

GENSIM differs from other process-oriented simulation systems in making a syntactic distinction between process preconditions that specify the *classes* of objects on which a process will execute, and those that specify the *properties* these objects must have for process execution [SimmonsM87]. I call these sets of preconditions the parameter-object classes and the preconditions (see Section 3.5.3. This distinction is important, and is worth making explicit for several reasons:

- Object types determine what processes are *activated*, and object properties and relationships determine which activated processes are *executed*. Making this distinction explicit emphasizes the difference between activation and execution to the programmer, and permits the process-activation optimizations described in Section 3.5.6.
- Intuitively, it is important for a person to know on what classes of objects a process operates.
- The process-specialization hierarchy within the process KB is isomorphic to the specialization hierarchy within the CKB. When we specialize a process, we often specialize the types of objects to which it applies. Separating type information from other object preconditions makes processes easier to specialize.

6.3.2 Process Granularity

One way to simplify process descriptions is to break different aspects of a complex physical interaction into different GENSIM processes, rather than describing the interaction with one complicated process. For example, although different processes describe different binding interactions between different objects, often the code that computes the filling of binding sites in two objects is not specific to the objects involved, so it seems natural to a programmer to isolate this code into a different process. A problem with this approach is that, in the interval between the execution of the object-specific process and that of the binding-site filling process, the GENSIM interpreter might execute another process on the objects in question. But the descriptions of the objects may be in an inconsistent state; for example, some slots may record that the objects are bound to one another when the object's binding sites are actually empty. Errors could easily result from having other processes operate on inconsistent object descriptions. A similar problem exists in production systems when a programmer attempts to separate common code from several large rules into one smaller rule. Three possible solutions are these:

- Use special slots to indicate when objects are in specific inconsistent states. All processes must check for such states. The problems associated with this approach are that it decreases process independence by forcing processes that describe unrelated physical events to check for inconsistencies caused by processes about which they should not have to know. These states would also lead to many wasteful process activations, and would increase system overhead.
- Do not define the binding-site filling code in a separate process, but use process inheritance to mix the code into each process that requires it. This approach is used in GENSIM; its drawback is that it makes process descriptions long and hard to read.

- Provide process "procedure calls" that allow one process to invoke another in a manner independent from the normal process interpreter. The problem with this approach is that it complicates reasoning about processes by adding an additional control mechanism to GENSIM.

6.3.3 Representational Tradeoffs in Process Preconditions

This section examines uses of the GENSIM process-description language that make it easier for us to share process preconditions among several processes; some of these constructs complicate the hypothesis-formation process.

It is useful to extend global scope to existentially quantified variables in process preconditions. For example, in the precondition

```
(EXISTS $x (AND (OBJECT.EXISTS $x 'OBJECT-TYPE)
                (IS.PART $x $y)))
```

the value of *\$x* normally would not be available to other expressions in the process preconditions and effects if the scoping of *\$x* in this process were interpreted strictly, so the scope of such variables is extended to all expressions in the process activation.

More generally, quantifiers are useful for writing general expressions that can be inserted into many processes, thus simplifying programming. They can be costly, however, during hypothesis generation. One example of this conflict arose in the early process precondition shown here. The precondition was intended to express the condition that, if any process parameter object contained a mutation, then the process should not fire:

```
(FORALL $param.obj
  (AND (MEMB $param.obj $Process.Parameters)
        (NOT (EXISTS $mutation
                  (AND (OBJECT.EXISTS $mutation
                        'Mutations)
                      (IS.PART $mutation
                                $param.obj)))))))
```

When designing hypotheses, HYPGENE often attempts to prevent processes from firing by violating their preconditions. HYPGENE attempted to violate the preceding precondition

process from firing by adding a new parameter object that contained a mutation to the process! Although not absolutely implausible, such hypotheses are more appropriately generated by operators that were explicitly designed to modify processes. But, removing the quantification requires a check for mutations within each parameter object.

A similar situation occurs with a newer precondition that is used to check for mutations in binding sites, and for affinities between binding sites; $\$x$ is the object containing the binding site:

```
(EXISTS $y (AND (OBJECT.EXISTS $y 'Binding.Sites)
                (IS.PART $y $x)
                (no mutation exists in $y)
                ($y has affinity for object $z)))
```

This condition is general because it can be mixed into the preconditions of any process to check for appropriate binding sites. The problem is that when HYPGENE's operators attempt to make this process fire, HYPGENE attempts to satisfy this existentially quantified condition by inserting every *Binding.Site* object in the SKB into $\$y$ — even if the $\$x$ object can contain only specific subclasses of binding sites. The solution is probably to give HYPGENE knowledge of what kinds of binding sites $\$x$ can have.

The conclusion here is that we must be careful when writing quantified expressions in processes because they can cause various sorts of problems when they are inverted. A topic for future research is to develop a theory that allows automated checking of expressions in process preconditions.

6.3.4 Predicate Calculus and Frames

A representational conflict existed throughout much of my work on hypothesis generation; namely a conflict between the use of a frame representation and use of a predicate-calculus representation for describing the SKB. I used a frame representation in GENSIM because that seemed the most natural way to represent the objects in this domain. The evolution of this

frame-based simulation system was discussed in Chapter 3. The frame representation works well in GENSIM for predicting experimental outcomes.

The need for predicate calculus became clear to me soon after I began work on HYPGENE. At first, it appeared possible that I could use frames to describe design goals — goals that describe objects that should or should not be present in P_A . We could simply mark those frames in P_A that should not exist, create new frames for objects that should exist, and treat these specifications as goals for the designer. However, when we consider using frames to represent the conditions that cause a process to fire or prevent a process from firing, it becomes clear that negation, disjunction, and quantification are indispensable representational primitives, because these concepts are notoriously difficult to represent using frames [Nilsson80]. Thus began the evolution of a hybrid system: GENSIM uses frames to represent the simulation state, but HYPGENE uses predicate calculus to represent design goals.

The next conflict arose when I realized that, since HYPGENE's execution of design operators alters the SKB, each design search state requires its own description of I_A' and P_A' . An ATMS seemed to be an appropriate tool for storing this information efficiently. Since KEE contains an ATMS, this ATMS was an obvious candidate [Intellicorp86]. As mentioned in Chapter 3, however, the functionality of KEEworlds is limited in that within different worlds (under different ATMS assumptions), it is impossible to introduce new objects or to delete existing objects; only slot values can be modified. This deficiency is serious, since the design operators frequently create and delete objects. Two choices became apparent: find a way of using KEEworlds, or use predicate calculus to represent the simulation state and use a more conventional ATMS to provide different versions of the SKB to different design states. I invested significant work in both alternatives, and found that, although both were possible, the latter required significantly more work because of the need to convert all the simulation LISP functions that manipulated frames to manipulate assertions.

I adapted KEEworlds to this task by adding a special slot to each object to indicate the

existence or nonexistence of the object in a particular world (new objects are created in the "background" or root world, and appear in every other world). Then I built a new layer of frame-access functions on top of the existing KEE functions that access the frame system.

6.4 Summary

We explored the implementation of HYPGENE in detail, considering such topics as how HYPGENE represents its design goals, how it expands search states, how it incrementally recomputes the prediction associated with a hypothesis, and how it detects when its design goals are satisfied.

This chapter also contained a thorough review of previous work on the problem of hypothesis formation. The work of different researchers differs along such dimensions as:

- The problem they attempt to solve — they might wish to compute a theory, the initial conditions of an experiment, a sequence of events, or some combination thereof
- The amount of starting information — their program might begin with a partial description of what it is to compute, for example, it might start with a buggy theory that it modifies; or the program might compute a theory from naught
- The problem-solving methods they employ — some researchers employ a heuristically guided generate-and-test procedure, whereas others formulate hypotheses by analyzing dependency records

We also discussed several empirical lessons of my work that involved the construction of process preconditions, and the use of predicate-calculus versus frame representations.

Chapter 7

Experiments

This chapter describes a number of experiments that I conducted with the GENSIM and HYPGENE programs (described in Chapters 3 and 5, respectively). These experiments serve four purposes. First, they demonstrate the claims that the GENSIM framework is sufficient to represent qualitative aspects of theories in molecular biology, and that the methods employed by HYPGENE are sufficient to solve hypothesis-formation problems in this domain. Second, the experiments clearly illustrate the capabilities of these programs by presenting examples of the programs' execution. Third, the experiments aid us in determining the strengths and weaknesses of these programs and of the methods behind them. The programs are sufficiently complex that it is not easy to predict how they will perform on even what appear to be simple test cases. Fourth, the experiments should make it clearer what knowledge of biology GENSIM has.

The first section of the chapter describes experiments in which GENSIM was used to predict the outcomes of several biological experiments. The second section describes experiments in which HYPGENE solved sample hypothesis-formation problems. Most of the problems used to test HYPGENE involve biological experiments that were test cases in the GENSIM section.

The word *experiment* is potentially ambiguous in this chapter because it can refer to both

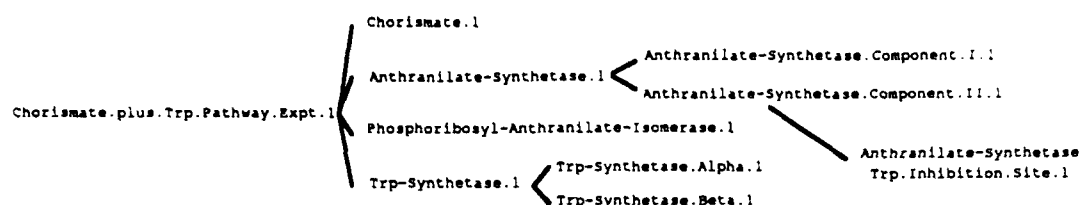


Figure 7.1: The initial conditions of the trp biosynthetic-pathway experiment. Every object in this figure contains the objects to its right as parts. For example, the Trp-Synthetase.1 enzyme has two parts: the alpha and beta subunits of the protein. The experiment as a whole is represented by the object Chorismate.plus.Trp.Pathway.Expt.1, which contains all the object in the experiment as parts.

computer-science experiments that involve GENSIM and HYPGENE, and to biological experiments. Henceforth, I use the word *trial* to refer to experiments with GENSIM and HYPGENE.

7.1 GENSIM Trials

In each trial of GENSIM I used the program to predict the outcome of a biological experiment. This section describes each trial by stating what objects were present in the initial conditions of the experiment whose outcome GENSIM predicts, and what reactions and new objects were predicted by GENSIM. For some trials, I show the internal structures of objects in the initial conditions or the prediction.

7.1.1 The Trp Biosynthetic Pathway

This simple trial models the trp biosynthetic pathway, in which a set of enzymes convert chorismate to tryptophan. The initial conditions of the experiment are shown in Figure 1.1: the predicted outcome is shown in Figure 1.2. Figure 1.3 shows the internal structure of every object in the prediction. GENSIM's prediction is correct in that it omits no reactions that should occur, it includes all reactions that do occur, and the objects produced by each reaction have the predicted parts and properties.

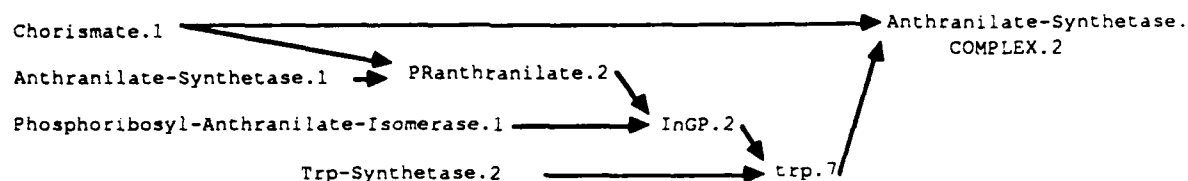


Figure 7.2: The predicted outcome of the trp biosynthetic-pathway experiment. The lines in this figure indicate the process firings whereby objects react to create the objects to their right. For example, Trp-Synthetase.2 reacts with InGP.2 to form trp.7.

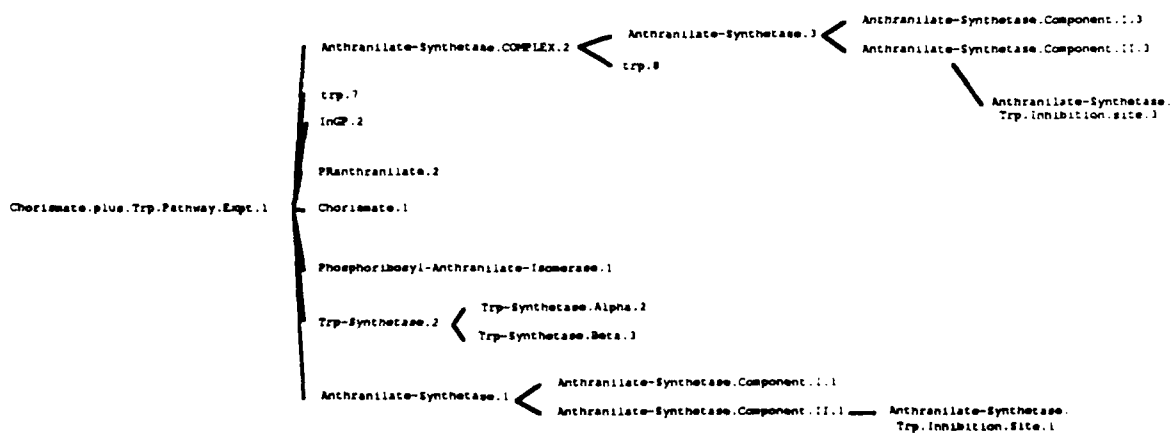


Figure 7.3: The internal structures of the objects predicted to be present at the end of the trp biosynthetic-pathway experiment.

7.1.2 The Trp Biosynthetic Pathway with a Mutant Enzyme

This trial is a variation of the previous trial. In this trial, the enzyme tryptophan synthetase contains a mutation that prevents it from catalyzing the reaction that converts InGP to tryptophan. The mutation is represented as an object that is part of the tryptophan-synthetase object. GENSIM correctly predicts that the last two (rightmost) reactions in Figure 1.2 do not occur.

7.1.3 Transcription of the Trp Leader Region

The leader-region transcription trial focuses on another subset of the overall trp system: the transcription of DNA. Figure 1.4 shows the objects in the initial conditions of this experiment, which include a truncated version of the trp operon called `Trp.Leader.Region.1` (I removed all the genes in the operon to make this trial easier to describe), the enzyme RNA polymerase, the trp aporepressor protein, and trp. The prediction generated by GENSIM is shown in Figure 1.5; this prediction is correct. The two sequences of reactions in this experiment fork the population of trp operon leader-region DNA into two classes: those whose operator regions bind to the activated repressor protein `Trp-Repressor.1`, and those whose promoters bind to `RNA-Polymerase.1` and undergo transcription to produce a messenger RNA.¹ The figures do not name `Trp.Leader.1` as participating in these reactions, but rather name the components of the operon that react: the promoter `Trp.Promoter.1` and the operator `Trp.Operator.1`.

The internal structure of one of the transcription-elongation complexes is shown in Figure 1.6. A transcription-elongation complex contains RNA polymerase, the DNA that RNA polymerase is transcribing, and the mRNA that RNA polymerase has synthesized thus far. The number of segments (parts) within the mRNA reflects the length of DNA that RNA polymerase has traversed. Since the mRNA contains two segments, we can infer that RNA

¹Most bacteria contain only a single copy of the trp operon. Thus, when I refer to the population of operons, I assume that the experiment takes place within a test tube that contains many cells, each of which has a trp operon.

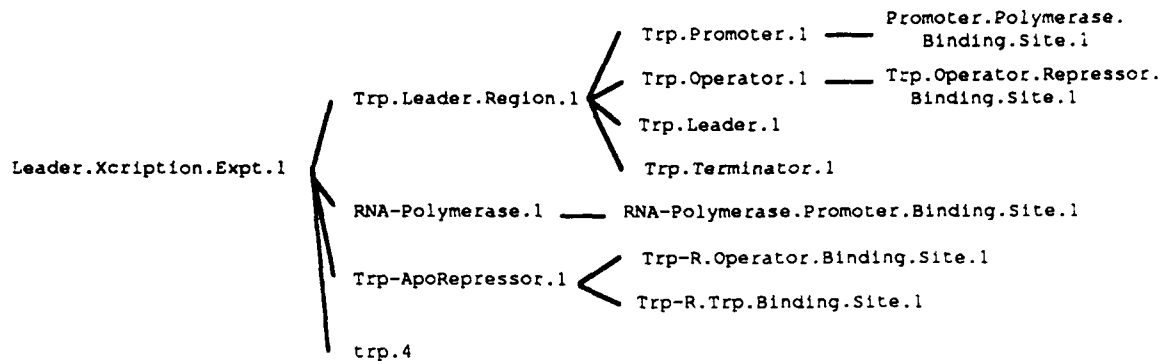


Figure 7.4: The objects in the initial conditions of the leader-region-transcription experiment.

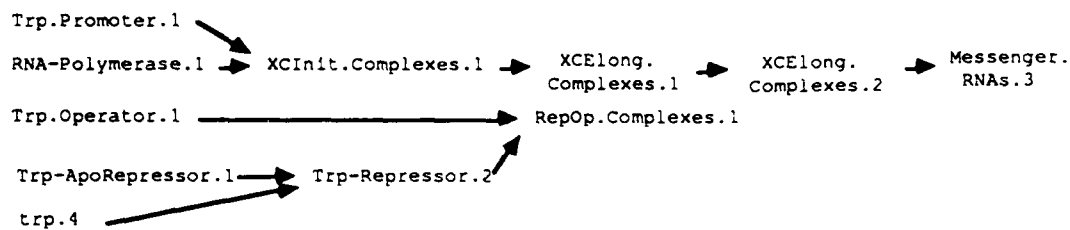


Figure 7.5: The outcome of the leader-region-transcription experiment that was predicted by GENSIM.

polymerase traveled two segments along the DNA to produce this transcription-elongation complex. Figure 1.7 shows the internal structures of every object in this experiment.

7.1.4 The Full Trp System

This trial simulates the entire trp system as it was known in the mid 1960s. Figure 1.8 shows the initial conditions of this experiment. GENSIM's prediction is shown in Figures ?? and ??. Figure ?? shows the transcription of the trp operon by RNA polymerase, which yields

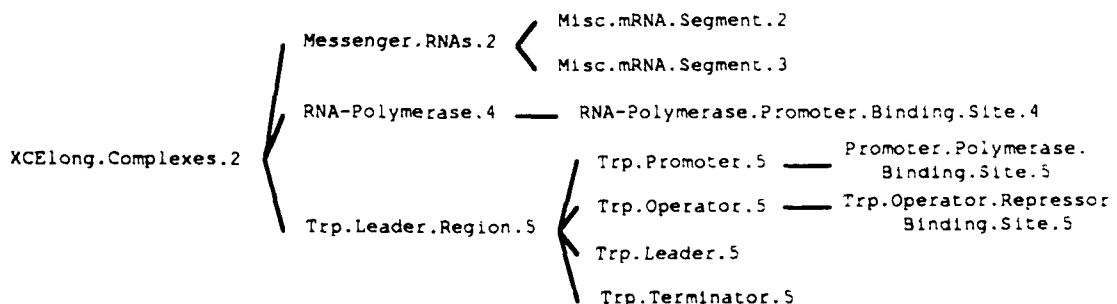


Figure 7.6: The internal structure of a transcription-elongation complex.

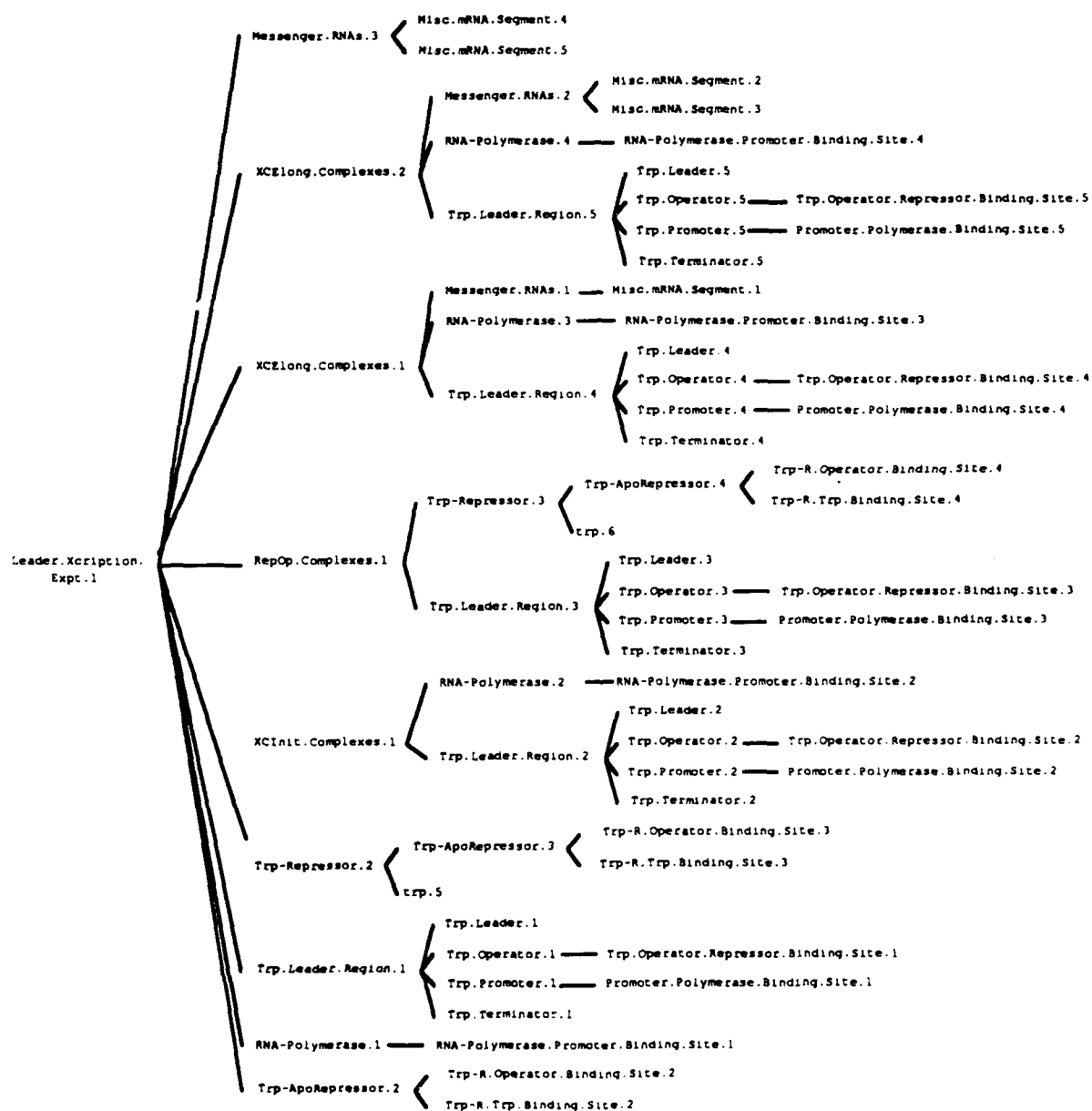


Figure 7.7: The objects in the predicted outcome of the leader-region-transcription experiment.

a free mRNA (**Messenger.RNAs.18**). Some of this mRNA is degraded into its constituent bases by the enzyme **RNase.1**. The mRNA also reacts with ribosomes, as shown in Figure ??.

Messenger.RNAs.18 contains five ribosome-binding sites, including **Ribosome.Binding.Site.47**.²

Each binding site attracts a ribosome, which translates the five different regions of mRNA into proteins such as **Trp-Synthetase.Beta.1**. Some of these proteins bind together to form larger, functional proteins, such as **Trp-Synthetase.1**.

The enzymes produced from translation of the trp-operon mRNA react with chorismate to carry out the steps in the trp pathway. The trp thus produced enters into several reactions: It binds to and inhibits anthranilate synthetase, and it activates the trp aporepressor protein (the latter complex then binds to the trp operator). Finally, the trp-tRNA-synthetase enzyme catalyzes the binding of tRNA^{trp} and trp to form charged tRNA^{trp} (which is used in all protein synthesis).

7.2 HYPGENE Trials

This section presents a number of trials of HYPGENE. Two of HYPGENE's inputs were the same for all these trials — the theory (process knowledge base), and the class knowledge base (CKB). The input that varied was the anomalous biological experiment presented to the program. In each trial HYPGENE's inputs included the initial conditions of an anomalous experiment (I_A), the outcome of the experiment predicted by GENSIM (P_A), and the difference between the predicted and observed outcomes of the experiment ($Error_A$). Most of these biological experiments are taken from the history of attenuation in Chapter 4, but I invented the first set of experiments. The experiments from the trp operon literature are from the early phases of the research on attenuation. For each trial I present the solutions computed by HYPGENE, and I discuss their completeness and correctness.

²I have modeled trp mRNA as it was known before Platt's experiments revealed the presence of a ribosome-binding site in the leader region of the operon [PlattY75].

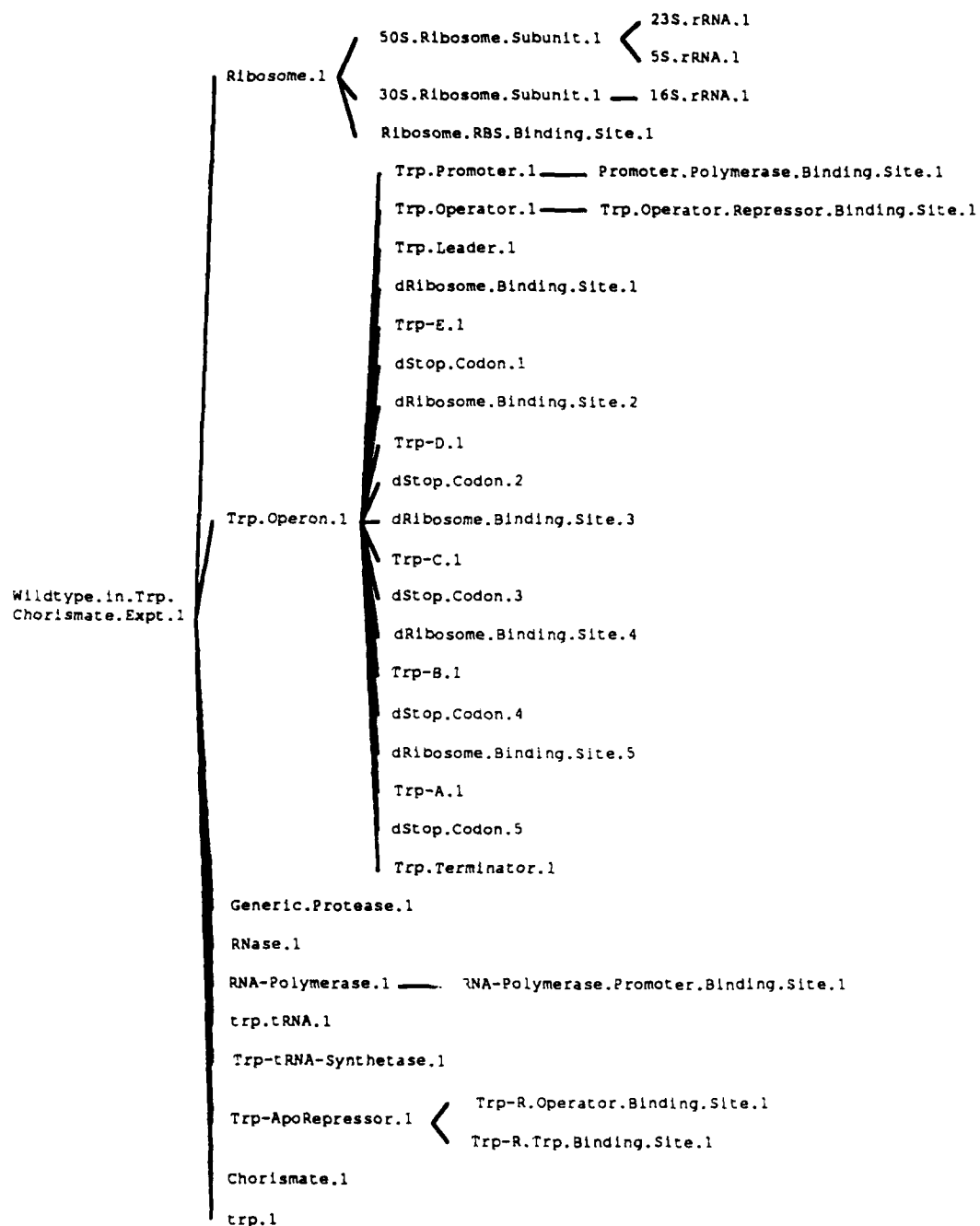


Figure 7.8: The initial conditions of the experiment involving the full trp system.

7.2.1 Trp Biosynthetic-Pathway Hypotheses

I invented the first set of HYPGENE trials; they involve the trp biosynthetic-pathway predictions from Section 1.1.¹ The goal of the first trial was for HYPGENE to identify elements of the pathway that, if missing or modified, would prevent the pathway from synthesizing trp. HYPGENE was presented with the experiment shown in Figures 1.1 and 1.2, and was given the design goal that the trp predicted by GENSIM should not in fact exist (Table ?? describes the predicates and functions used in these goals and hypotheses):

(NOT (OBJECT.EXISTS 'trp.7 'trp))

HYPGENE designed three types of hypotheses:

- Certain objects or parts of objects were missing from I_A , such as chorismate and the enzyme anthranilate synthetase
- Enzymes or enzyme-binding sites contained mutations that interfered with the activity of the enzyme
- Trp was bound to a site within the enzyme anthranilate synthetase; in this state, the enzyme cannot catalyze the first reaction in the pathway

The actual hypotheses are shown in Figure 1.9. These hypotheses are both complete and correct in the sense that Dr. Yanofsky verified that HYPGENE had neither missed any solutions, nor generated any solutions that are wrong.

The next trial involved generation of quantitative hypotheses. The same simulation of the trp biosynthetic pathway was presented to HYPGENE, with the goal of decreasing the amount of trp present in the experiment:

(Decrease.Quantity 'trp.7)

Once again, HYPGENE generated three different types of hypotheses:

```

(RETRACT (OBJECT.EXISTS 'Chorismate.1 'Chorismate))

(RETRACT (OBJECT.EXISTS 'Anthranilate-Synthetase.1 'Anthranilate-Synthetase))

[ASSERT (MEMB 'trp.69
              (W/GET.VALUES 'Anthranilate-Synthetase.Trp.Inhibition.Site.1
                           Object.Interacting.With.Site))]

(CREATE.OBJECT 'trp.69 'trp)
(BINDV $class 'trp)

(RETRACT (OBJECT.EXISTS 'Anthranilate-Synthetase.Trp.Inhibition.Site.1
                        'Active.Sites))

[ASSERT (MEMB 'Anthranilate-Synthetase.Catalysis
              (W/GET.VALUES 'Mutations.225 'Processes.Disabled))]
(ADD.PART 'Mutations.225
          'Anthranilate-Synthetase.Trp.Inhibition.Site.1)
(CREATE.OBJECT 'Mutations.225 'Mutations)

[ASSERT (MEMB 'Anthranilate-Synthetase.Catalysis
              (W/GET.VALUES 'Mutations.224 'Processes.Disabled))]
(ASSERT (IS.PART 'Mutations.224
                 'Anthranilate-Synthetase.Trp.Inhibition.Site.1))
(CREATE.OBJECT 'Mutations.224 'Mutations)

[ASSERT (MEMB 'Anthranilate-Synthetase.Catalysis
              (W/GET.VALUES 'Mutations.223 'Processes.Disabled))]
(ASSERT (IS.PART 'Mutations.223 'Anthranilate-Synthetase.1))
(CREATE.OBJECT 'Mutations.223 'Mutations)

(RETRACT (OBJECT.EXISTS 'Phosphoribosyl-Anthranilate-Isomerase.1
                        'Phosphoribosyl-Anthranilate-Isomerase))

[ASSERT (MEMB 'Phosphoribosyl-Anthranilate-Isomerase.Catalysis
              (W/GET.VALUES 'Mutations.222 'Processes.Disabled))]
(ASSERT (IS.PART 'Mutations.222
                 'Phosphoribosyl-Anthranilate-Isomerase.1))
(CREATE.OBJECT 'Mutations.222 'Mutations)

(RETRACT (OBJECT.EXISTS 'Trp-Synthetase.2 'Trp-Synthetase))

[ASSERT (MEMB 'Trp-Synthetase.Catalysis
              (W/GET.VALUES 'Mutations.221 'Processes.Disabled))]
(ASSERT (IS.PART 'Mutations.221 'Trp-Synthetase.2))
(CREATE.OBJECT 'Mutations.221 'Mutations)

```

Figure 7.9: Hypotheses formulated by HYPGENE to account for the absence of trp in an experiment involving the trp biosynthetic pathway. Eleven hypotheses are shown. Each hypothesis consists of one or more changes to I_A . For example, the first hypothesis retracts the assertion that the object chorismate was present in I_A .

```

(CREATE.OBJECT 'Trp-tRNA-Synthetase.2 'Trp-tRNA-Synthetase)
(CREATE.OBJECT 'trp.tRNA.3 'trp.tRNA)

(CREATE.OBJECT 'Trp-ApoRepressor.2 'Trp-ApoRepressor)

(DECREASE.INTRINSIC-RATE 'Trp-Synthetase.Catalysis)

(DECREASE.STARTING Trp-Synthetase.2)

(CREATE.OBJECT 'Trp-Synthetase.Alpha.3 'Trp-Synthetase.Alpha)

(DECREASE.INTRINSIC-RATE
  'Phosphoribosyl-Anthranilate-Isomerase.Catalysis)

(DECREASE.STARTING
  'Phosphoribosyl-Anthranilate-Isomerase.1)

(DECREASE.INTRINSIC-RATE
  'Anthranilate-Synthetase.Catalysis)

(DECREASE.STARTING 'Anthranilate-Synthetase.1)

(DECREASE.STARTING 'Chorismate.1)

```

Figure 7.10: Hypotheses formulated by HYPGENE to account for decreased levels of trp in an experiment involving the trp biosynthetic pathway. The first hypothesis proposes that trp-tRNA-synthetase and tRNA^{trp} were present in I_A ; HYPGENE knows that these objects react with trp.

- The intrinsic rate of each process in the pathway was decreased
- The starting amounts of each enzyme, and of chorismate, were decreased
- Other objects were present in I_A that reacted with and consumed trp

HYPGENE's hypotheses are shown in Figure 1.10. These hypotheses are correct and complete, although one hypothesis is only marginally correct. HYPGENE proposes that the trp aporepressor is present in I_A , because it knows that this object reacts with trp. HYPGENE does not know, however, that the concentration of the aporepressor protein is usually so small relative to trp that this reaction would have little effect on the concentration of trp. This type of analysis requires knowledge of the normal concentrations of these objects, and of the amount of decrease in trp for which this hypothesis must account.

The results of two similar trials are not shown. In the first, HYPGENE was instructed to generate hypotheses to increase the quantity of trp in the biosynthetic-pathway experiment; it

produced hypotheses analogous to those in Figure 1.10. The second trial involved a subset of the trp pathway. Because this subset lacked the final enzyme — tryptophan synthetase — no trp was produced by the pathway. HYPGENE's goal was to explain what factors could account for the observation of trp:

(EXISTS \$trp (OBJECT.EXISTS \$trp 'trp))

HYPGENE correctly proposed that either trp-synthetase or its two components (which react to form the enzyme itself) were present in I_A .

7.2.2 Transcription-Initiation Hypotheses

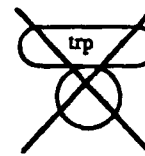
This set of trials involves a transcription-initiation experiment similar to that described in Section 1.1.3, except that this experiment contains the full trp operon. GENSIM's prediction is shown in Figure 1.5. The problem considered here is a typical examination question posed to undergraduate students in an introductory course in molecular biology, and was addressed by Hiraga in his early studies of the trp system [Hiraga69]. HYPGENE was presented with the simulation in Figure 1.5 and was given the design goal

(FORALL \$obj (NOT (OBJECT.EXISTS \$obj 'RepOp.Complexes)))

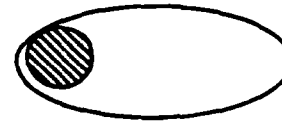
This goal directs HYPGENE to formulate hypotheses to explain why the cell's repression mechanism might fail to produce a repressor-operator complex. Prediction errors of this sort usually are caused by mutations in various objects in the cell.

HYPGENE produced 15 solutions to this problem; the solutions fall into the five different classes that are diagrammed in Figure 1.11; the actual solutions are shown in Figure 1.12. Section 5.3 contains a detailed trace of the reasoning HYPGENE used to derive one of these solutions. These hypotheses are complete and correct. I anticipated classes 1 and 4 as the obvious types of hypotheses to be produced; I had not anticipated classes 2, 3, or 5. In a sense, classes 2 and 5 are redundant with class 4, because mutations would be the biological

Class 1: Postulate that certain objects were in fact not present in the initial conditions of the experiment



Class 2: Postulate that binding sites were absent from specific objects



Class 3: Postulate that specific binding sites were occupied, and thus were prohibited from participating in reactions



Class 4: Postulate that specific binding sites contained mutations that had specificities that interfered with the reactivity of the binding sites



Class 5: Postulate that the specificities of binding sites were altered in a manner that prohibited them from participating in reactions

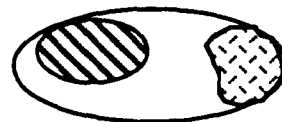


Figure 7.11: The different classes of hypotheses formulated by HYPGENE to account for the absence of repressor-operator complexes in a transcription-initiation experiment.

cause of the changes in 2 and 5. HYPGENE does not have enough general knowledge about mutations to recognize this redundancy. We could argue that classes 2 and 5 are byproducts of the programming style that I used to write process preconditions. I often wrote existentially quantified expressions to identify a binding site of a given specificity within an object, and then wrote expressions that determined whether the binding site was empty and free of mutations. HYPGENE prevents processes from firing by negating their preconditions; when this type of reasoning negates existentially quantified expressions, it yields the hypothesis that no binding site of the required specificity existed within the object.

```

(RETRACT (OBJECT.EXISTS 'Trp.Operator.1 'Trp.Operator))

(RETRACT (OBJECT.EXISTS 'Trp-ApoRepressor.2 'Trp-ApoRepressor))

(RETRACT (OBJECT.EXISTS 'trp.4 'trp))

(RETRACT (OBJECT.EXISTS 'Trp.Operator.Repressor.Binding.Site.1 'Active.Sites))

[RETRACT (MEMB 'Trp-Repressor
              (W/GET.VALUES 'Trp.Operator.Repressor.Binding.Site.1
                           'Potential.Interacting.Objects)]

(RETRACT (OBJECT.EXISTS 'Trp-R.Operator.Binding.Site.2
                        'Active.Sites))

[RETRACT (MEMB 'Trp.Operator
              (W/GET.VALUES 'Trp-R.Operator.Binding.Site.2
                           'Potential.Interacting.Objects)]

(RETRACT (OBJECT.EXISTS 'Trp-R.Trp.Binding.Site.2 'Active.Sites))

[RETRACT (MEMB 'trp
              (W/GET.VALUES 'Trp-R.Trp.Binding.Site.2
                           'Potential.Interacting.Objects)]

[ASSERT (MEMB 'Trp-Repressor.17
             (W/GET.VALUES 'Trp.Operator.Repressor.Binding.Site.1
                          'Object.Interacting.With.Site)]

(ASSERT (OBJECT.EXISTS 'Trp-Repressor.17 'Trp-Repressor))

[ASSERT (MEMB 'trp.4
             (W/GET.VALUES 'Trp-R.Trp.Binding.Site.2
                          'Object.Interacting.With.Site)]

[ASSERT (MEMB 'trp.23
             (W/GET.VALUES 'Trp-R.Trp.Binding.Site.2
                          'Object.Interacting.With.Site)]

(ASSERT (OBJECT.EXISTS 'trp.23 'trp))

(ASSERT (IS.PART 'Mutations.13
                'Trp.Operator.Repressor.Binding.Site.1))

[ASSERT (MEMB 'Trp-Repressor.Binds.Operator
             (W/GET.VALUES 'Mutations.13 'Processes.Disabled)]

(ASSERT (OBJECT.EXISTS 'Mutations.13 'Mutations))

(ASSERT (IS.PART 'Mutations.17
                'Trp-R.Operator.Binding.Site.2))

[ASSERT (MEMB 'Trp-Repressor.Binds.Operator
             (W/GET.VALUES 'Mutations.17 'Processes.Disabled)]

(ASSERT (OBJECT.EXISTS 'Mutations.17 'Mutations))

(ASSERT (IS.PART 'Mutations.19 'Trp-R.Trp.Binding.Site.2))

[ASSERT (MEMB 'Trp-ApoRepressor.Binds.Trp
             (W/GET.VALUES 'Mutations.19 'Processes.Disabled)]

(ASSERT (OBJECT.EXISTS 'Mutations.19 'Mutations))

```

Figure 7.12: Hypotheses formulated by HYPGENE to account for the absence of repressor-operator complexes in a transcription-initiation experiment. The last hypothesis proposes that a mutation object exists, that the mutation is part of the binding site within the trp-repressor protein that binds to trp, and that the functionality of this mutation is such that it interferes with the process `Trp-ApoRepressor.Binds.Trp`.

An interesting observation from this trial is that HYPGENE's performance would be improved if the program possessed more knowledge about the allowed configurations of biological objects. In an earlier version of HYPGENE, one of the hypotheses generated for this trial prevented processes from firing by violating a process precondition of the form (IS.PART X Y) where X was a binding site and Y was an enzyme. It did so by asserting that X was no longer a component of Y . This hypothesis is not reasonable, because we do not observe disembodied protein binding-sites floating free in the cell. Thus, a special slot was added to each object to indicate whether or not the object could be found free in solution. In other problems, HYPGENE generates similar types of hypotheses, such as proposing that one object is a component of another, or that a slot contains additional values. Although biologists have extensive knowledge that could be used to evaluate these types of hypotheses, HYPGENE does not currently incorporate such knowledge.

7.2.3 Attenuation Hypotheses

The experiments described in [JacksonY73] were among the most important experiments involved in the discovery of attenuation. The hypothesis-formation problems resulting from these experiments posed the most difficult challenge HYPGENE faced.

This trial involved a key pair of experiments from [JacksonY73] that compared the rate of trp-mRNA production in two different *E. coli* strains. The reference strain (E_R) had a nonfunctional trp-repressor protein; the anomalous strain (E_A) had both a nonfunctional trp-repressor and a deletion in the leader region of the trp operon. CENSIM predicts the same outcome for both experiments, which is shown in Figure 1.13. The observed outcome of the experiment involving the second strain was anomalous because the observed concentration of trp-mRNA was higher for this strain than for the first strain, whereas theory predicts that the concentrations should be the same.

As discussed in Section 4.4.3, Jackson and Yanofsky proposed several hypotheses to account

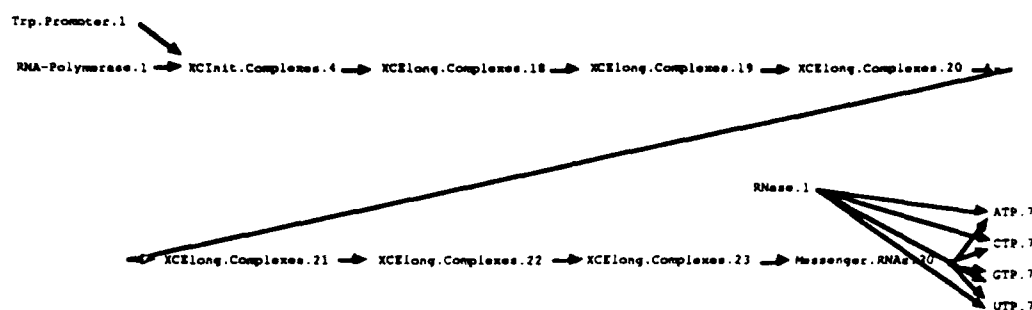


Figure 7.13: The outcome of the Jackson-Yanofsky experiment as predicted by GENSIM.

for the discrepancy in trp-mRNA concentrations:

1. The second strain contained multiple copies of the trp operon, yielding a higher rate of transcription
2. The second strain degraded trp-mRNA at a rate lower than that of the first strain
3. The gene-splicing technique that they had used to create the leader-region deletion in the second strain had created a second promoter in the leader region, yielding a higher rate of transcription
4. The leader region contained a site that decreased mRNA production in the first strain; this site was removed by the deletion, yielding an increased rate of transcription in the second strain (later experiments showed that this hypothesis was the correct one; it describes attenuation)

HYPGENE was given the goal of increasing the amount of mRNA present:

`(Increase.Quantity 'Messenger.RNAs.17)`

HYPGENE exhibited good, but not perfect, performance on this problem. Its solutions are shown in Figure 1.14. HYPGENE formulated all the hypotheses in the preceding list except for hypothesis 3; later in this section we consider why it missed that hypothesis. HYPGENE also formulated additional hypotheses that the biologists did not propose. Most of these hypotheses

```

(INCREASE.INTRINSIC.RATE RNase.Catalysis)
(DECREASE.STARTING Trp.Promoter.1)
(DECREASE.STARTING RNA-Polymerase.1)
(DECREASE.INTRINSIC.RATE Polymerase.Binds.Promoter)
(DECREASE.INTRINSIC.RATE Transcription.Initiation)
(DECREASE.INTRINSIC.RATE Transcription.Elongation)
(DECREASE.INTRINSIC.RATE Transcription.Termination)
(ASSERT (OBJECT.EXISTS 'Trp.Leader.ED102.1
                        'Leaky.Terminators))

```

Figure 7.14: Hypotheses formulated by HYPGENE to account for the unexpectedly high expression of the trp operon in the Jackson-Yanofsky experiment.

were then rejected by HYPGENE when it used one of the hypothesis-filtering mechanisms associated with reference experiments. Later in this section we consider how those hypotheses were filtered, and identify additional knowledge that can be used to reject the remaining extra hypotheses. First, we consider how HYPGENE produced hypothesis 4.

Formulation of Hypothesis 4

The first time HYPGENE was run on this trial, it did not produce hypothesis 4 (page 16) because of an omission in HYPGENE's general methods for generating quantitative hypotheses. I formulated these methods to generate quantitative hypotheses for a *single* experiment, but the experiment from [JacksonY73] involves accounting for a *relative* increase in the mRNA present in *two* experiments, where the bacterium present in E_A is very similar to that present in E_R . In this situation, we can generate a whole new set of hypotheses by "inverting" the normal use of HYPGENE's quantitative operators. That is, to generate hypothesis 4, instead of directly generating hypotheses that would *increase* the mRNA in E_A , we generate hypotheses that would *decrease* the mRNA in E_R , and postulate that the leader-region deletion in E_A removed whatever HYPGENE proposed had decreased the mRNA concentration in E_R , thus *increasing* the relative amount of mRNA in E_A .

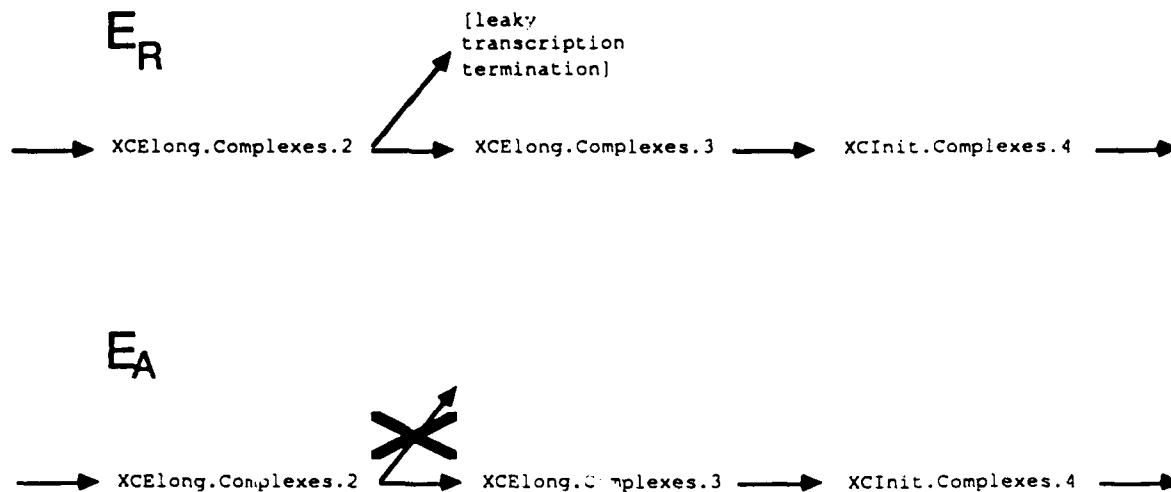


Figure 7.15: One of HYPGENE's solutions to the Jackson-Yanofsky experiment — hypothesis 4. HYPGENE accounted for an increase in trp concentration in E_A by postulating that a leaky transcription termination process was decreasing the rate of trp-mRNA synthesis in E_R , but that this process was not occurring in E_A .

HYPGENE's operator `Add.Consumption.Path.To.Dec.Quantity` formulated this hypothesis by proposing that a process called leaky transcription termination occurs in E_R , thus diverting away some of the transcription elongation complexes that eventually produce a full-length trp-mRNA (see Figure 1.15). HYPGENE also proposes that this additional reaction is *missing* in E_A , thus removing the diversion of transcription-elongation complexes, and increasing the amount of mRNA in E_A relative to E_R . Note that this hypothesis asserts that the additional process in E_R had been present all along, so the rate of transcription initiation must have been higher than they had thought to have yielded the observed amount of mRNA — the earlier model said that all transcription-initiation complexes yield trp-mRNA, but this model says that only some of the transcription-initiation complexes transcribe the full length of the trp operon. (HYPGENE does not make this deduction about the rate of transcription initiation.)

Three other points are of interest for this hypothesis. First, to make the leaky transcription termination process fire in E_R , HYPGENE postulates that the leader region of the operon contains a leaky transcription terminator. This hypothesis captures one-half of the properties of the attenuator region as it was formulated by the trp researchers; the researchers also proposed that the rate of attenuation was dependent on trp concentration, as shown by later

experiments [BertrandY76]. Second, note that HYPGENE did not formulate the attenuator out of thin air; HYPGENE combined the existing trp operon with the existing concept of the leaky transcription terminator. The CKB contains descriptions of both leaky and nonleaky transcription termination sites, and the PKB describes processes of both leaky and nonleaky transcription termination (the difference being that *both* leaky transcription termination and transcription elongation can occur at a leaky transcription terminator, whereas only nonleaky transcription termination can occur at a nonleaky terminator). Professor Yanofsky confirms that this knowledge is historically accurate; in the early 1970s biologists believed that both types of terminators might exist — they had not shown whether or not transcription terminators were leaky, and they acknowledged both possibilities [Yanofsky89].

The third point is that HYPGENE attempts to formulate another class of hypotheses involving a terminator in the leader region rather than a leaky terminator. We reject these hypotheses because they predict that *no* trp-mRNA would be produced by the trp operon (since *all* transcription elongation halts at the nonleaky terminator).

The Missing Hypothesis

HYPGENE missed hypothesis 3 because this hypothesis could not be generated from the goal (Increase.Quantity 'Messenger.RNAs.17). The reason is that hypothesis 3 does not actually satisfy this goal. Hypothesis 3 proposes that the deletion of the leader region coincidentally produced a new promoter in the trp operon where the edges of the operon that bordered on the deletion were spliced together (biologically, this is improbable, but possible). The new promoter would produce an mRNA that is similar to, but slightly shorter than, the normal mRNA produced by the trp operon (Messenger.RNAs.17). Strictly speaking, if the cell produced more of this new mRNA, the cell would *not* increase the quantity of Messenger.RNAs.17. However, the experimental techniques that Jackson and Yanofsky used to detect mRNA would not distinguish the shorter mRNA from Messenger.RNAs.17, so the biologists's measurements

did indicate an increase in *trp*-mRNA.

HYPGENE must be given a slightly different design goal if it is to generate this hypothesis: namely, it must have the goal of increasing the quantity of either *Messenger.RNAs.17* or any other mRNA that would be indistinguishable from *Messenger.RNAs.17*, *given the experimental techniques in use for detecting mRNA*. To satisfy this goal, HYPGENE would need knowledge of the techniques that the biologists used to measure mRNA, and of what mRNA species these techniques would and would not be able to distinguish. This knowledge is beyond the scope of this thesis.

The Extra Hypotheses

HYPGENE generated several hypotheses that were not proposed by the biologists. One of these hypotheses explains the increased mRNA in E_A by postulating that the level of RNA polymerase was elevated in I_A . Because RNA polymerase is an input to the reaction network in Figure 1.5, this hypothesis appears to be reasonable; the biologists did not propose it because they knew that RNA polymerase is generally present in excess in the cell; that is, increasing the concentration of RNA polymerase will not alter the rate of transcription initiation. This type of knowledge could easily be added to HYPGENE using the framework developed in Section 3.4.1. Similarly, HYPGENE proposed that the rates of the transcription initiation, elongation, and termination processes, and that of the polymerase-binds-promoter process, were decreased. The biologists ruled out these explanations because previous studies of transcription showed that the rates of all these processes were generally constant [Yanofsky89].

The other extra hypotheses produced by HYPGENE were similar to hypothesis 3 (page 16). Hypothesis 3 proposes that a leaky transcription terminator existed in the leader region. As HYPGENE worked backward through the reaction network in Figure 1.13, attempting to find an explanation for the increased concentration of *Messenger.RNAs.17*, it created hypotheses that propose that the leaky transcription terminator lies within every DNA segment within the

trp operon, such as the gene *TrpA*. 1 in Figure 1.6. The biologists ruled out these hypotheses because the deletion occurred in the leader region and thus the deletion could remove only a leaky transcription terminator that was present in the leader region, and not, for example, one that was in *TrpA*. 1 (as shown in Figure 1.15, E_R must contain the leaky transcription terminator, whereas E_A must not contain it). I discussed this type of reasoning in Section 5.8.4. To filter the hypotheses generated to account for the anomalous outcome of E_A , we can use knowledge of what changes to the initial experimental conditions can be caused by the experimental techniques used to make I_A differ from I_R . In this example, a gene-splicing technique deleted a region of DNA from I_R to yield I_A . HYPGENE generated a set of hypotheses that postulated that a leaky transcription terminator was present in various regions of the trp operon in I_R , but that no such leaky transcription terminator existed in I_A . The gene-splicing technique could have removed the leaky transcription terminator only if the terminator existed in the leader region of the operon, so all other proposed locations for the leaky transcription terminator were rejected (HYPGENE used a simple theory of gene splicing to prune away all these hypotheses, except for hypothesis 3).

7.3 Summary

Both HYPGENE and GENSIM have solved a number of problems from the history of attenuation. GENSIM produced flawless predictions of the outcomes of experiments involving the trp biosynthetic pathway, the transcription of the trp operon, and the entire trp operon gene-regulation system. HYPGENE produced correct solutions to hypothesis-formation problems involving both the trp biosynthetic pathway and the repression of the trp operon. When applied to a more difficult pair of experiments from [JacksonY73], HYPGENE produced all but one of the hypotheses proposed by Jackson and Yanofsky, and formulated several other hypotheses

that Jackson and Yanofsky did not propose. To formulate the hypothesis that it missed, HYPGENE would require knowledge of laboratory techniques that is beyond the scope of this thesis. Most of the extra hypotheses that HYPGENE generated were filtered by information from a reference experiment, using the techniques described in Section 5.8.4. The biologists did not propose the extra hypotheses that HYPGENE generated because they possessed quantitative knowledge that HYPGENE did not have, but that could be represented using the techniques described in Section 3.4.

Chapter 8

Conclusions

This chapter analyzes and summarizes the thesis. We assess the weaknesses and limitations of the techniques presented in the preceding chapters, and consider the assumptions on which these techniques depend. We also evaluate the contributions of this work.

8.1 The Historical Study of Attenuation

Chapter 4 presented a historical study of the discovery of a bacterial gene-regulation mechanism called attenuation. The study described how the theory of the regulation of the *trp* operon evolved over time. It discussed experiments that the biologists performed, and presented the alternative hypotheses that the biologists formulated to explain the outcomes of anomalous experiments.

8.1.1 Limitations

The principal limitation of the historical study is methodological: the historical records that I studied consisted of publications in scientific journals and interviews with the biologists who performed the research. I did not examine the biologists' laboratory notebooks, which historians of science consider to be an important source of information. Although this omission

may have affected the accuracy of the study, it is important to note that many historians of science study research that was performed so long ago that they cannot interview the scientists involved. The use of interviews may compensate for the omission of notebooks.

8.1.2 Contributions

The main contributions of this study result from its size and complexity. The trp-operon research required over 50 person-years of work spanning 15 years. The study shows a long progression of modern scientific research, rather than an isolated snapshot of work. It presents a balanced view of the different kinds of problems that molecular biologists solve. We see the context in which different research problems arise. Because I undertook it so soon after the biological research was performed, the study is particularly detailed, and, I believe, accurate.

The study provided a testbed of examples for the methods developed in this dissertation; these examples could also be useful to future researchers in AI, and in the philosophy and history of science.

My analysis of the study yielded two results: First, I proposed a set of modes of scientific exploration that classify scientific experiments, and that identify the factors that scientists consider when determining what types of experiments to perform next. These modes are confirmation, discrimination, theory generation, fact finding, and technique development. Second, I proposed a set of theory-modification operators that describe the different types of syntactic changes that the biologists made to their theories of the trp operon. I refined these operators to yield the hypothesis-design operators that form the core of the HYPGENE program.

8.2 Declarative Device Modeling

Chapter 3 presented methods for constructing declarative device models of the trp operon. Model 2 focuses on quantitative aspects of the trp operon, such as chemical reaction rates and object concentrations. Model 3 (GENSIM) is concerned with describing the structures and

properties of the objects in experiments involving the trp operon, and with predicting what chemical reactions occur among these objects.

8.2.1 Limitations

The strengths of Model 2 are the weaknesses of Model 3, and viceversa. Model 3 does not reason about quantitative aspects of the trp operon. Model 2 neither reasons about the structures of the objects whose concentrations it represents, nor predicts the occurrence of the reactions for which it computes the rates. An obvious goal for future research is to combine these two models.

Neither Model 2 nor model 3 reasons about temporal or spatial aspects of the trp operon. Although many interesting problems in this domain do not involve time or space, many do.

GENSIM incorporates the assumption that its predictions span a sufficiently short time interval that no population of objects within a simulation will be fully depleted. If GENSIM were to reason about temporal aspects of the regulation of the trp operon, this assumption would be violated.

8.2.2 Contributions

Model 2 describes new qualitative representations for describing both state-variable values and mathematical dependencies among state variables. These representations allow us to use the knowledge that biologists have about a biological system, be that knowledge precise or imprecise. We can assign precise quantitative values to state variables, or we can make assertions that constrain a variable's value more generally. Users can describe mathematical dependencies using frames called functions, pairwise interactions, and mappings. Finally, we considered several classes of inference that can be used to propagate variable values through these different types of mathematical dependencies.

The GENSIM model contributes a number of new simulation techniques. Its class KB

describes different types of biological objects that we can instantiate in an experiment-specific KB to describe the objects in the initial conditions of an experiment. The process KB describes both particular chemical reactions and general classes of chemical reactions; it also uses KEE's frame inheritance to define processes in a novel way. This use of inheritance is applicable to traditional production-rule languages as well as to GENSIM's process-description language. I explored a number of issues involved in managing objects in GENSIM simulations, including the conditions under which we must copy-then-modify objects rather than modifying them directly, the benefits of merging the descriptions of identical objects that are created during a simulation, and a method for sharing descriptions of similar objects in a simulation using an ATMS. I also presented the algorithm that GENSIM uses to activate processes efficiently.

Although I developed the models in Chapter 3 in the domain of molecular biology, I believe that the methods used to construct these models are applicable to other domains. These methods refer to general concepts such as system-state variables, mathematical dependencies among state variables, object part-whole structures, and processes with preconditions and effects.

8.3 Hypothesis Formation

Chapters 5 and 6 described how we can use design and planning methods to solve hypothesis-formation problems. A hypothesis-formation problem occurs when the predicted outcome of an experiment conflicts with the observed outcome. The design framework instructs us to treat the elimination of the prediction error as a design goal. We employ design operators to satisfy this goal. The operators work backward from the prediction error; they formulate modifications to the chemical-reaction theory and to what were believed to be the initial conditions of the experiment, to align prediction with observation.

8.3.1 Limitations

The design framework assumes that HYPGENE's inputs exist, and that they have certain properties. These assumptions are violated by scientific-reasoning problems that require us to derive one or more of the entities that HYPGENE takes as an input, such as a detailed description of the initial conditions of the experiment. For example, BACON computes theories from naught. In addition, the descriptions of I_A and T that we supply to HYPGENE and GENSIM must allow GENSIM to compute a predicted outcome for the experiment; DENDRAL cannot predict an experimental outcome from its partial description of I_A .

Another assumption of this method is that, to satisfy domain-specific goals such as (IS .PART X Y), HYPGENE depends on the existence of a LISP function that knows how to invert the predicate — to make X a component of Y . Such functions are easy to define for predicates that have a unique inverse. (Section 6.2.7 described how this assumption is violated in DENDRAL's domain.)

HYPGENE has a limited ability to regress design goals through process effects to determine what SKB contents will cause a process to achieve a desired goal. Some processes call recursive LISP procedures, which are notoriously difficult for a program to reason about. AI researchers have not developed a general theory of goal regression, but HYPGENE needs one. The lack of such a theory has not been a problem for the test cases we have run, because HYPGENE contains heuristics (summarized in Section 5.5.1) to solve a limited set of goal-regression problems.

HYPGENE's design operators reflect the GENSIM assumption that process effects cannot delete objects from the simulation (see Section ??). If we were to extend GENSIM's qualitative chemistry such that it no longer incorporates this assumption, we would have to augment HYPGENE's operators; for example, by creating an initial-condition design operator that attempts to remove an assertion from a prediction by firing an existing process that removes the assertion.

8.3.2 Contributions

HYPGENE advances the state of the art of hypothesis formation because it has solved complex scientific problems from the real-world domain of molecular biology. The realism of the problems that HYPGENE has solved was thoroughly documented in the historical study in Chapter 4, and in the trials in Chapter 7. The complexity of the domain theories that HYPGENE manipulates was described in Chapter 3, in the trials in Chapter 7, and in Appendices A and B.

This dissertation has shown that it is profitable to treat hypothesis-formation as a design problem. Design provides both an interesting metaphor for thinking about hypothesis-formation problems, and specific methods for solving these problems. Chapter 5 presented an overview of these methods;¹ it gave a precise statement of the hypothesis-formation problem, and described how a hypothesis-formation problem is phrased as a design goal. It then presented HYPGENE's design operators: the initial condition design operators, quantitative-hypothesis design operators, process-design operators, and class-KB design operators (the latter two types of operators were not implemented). Next the chapter discussed the computational complexity of HYPGENE's search, and considered several criteria that could be used to control this search: simplicity, domain knowledge, operator-precedence information, and reference experiments.

One type of operator-precedence information that I observed from the historical study of attenuation is that biologists generally prefer to define new processes by instantiating an existing process class, rather than by making arbitrary syntactic changes to an existing process. Chapter 5 described three methods for using reference experiments to evaluate hypotheses and to guide the hypothesis-formation process. A reference experiment has initial conditions that are similar to those of the anomalous experiment, but the outcome of the reference experiment

¹Sections 5.2 and 1.3 mentioned other design methods that I have not yet applied to hypothesis-formation problems.

is predicted correctly by theory. First, we can reject hypotheses that would alter the predicted outcome of the reference experiment, P_R , since P_R was already correct. Second, we can reject hypotheses that do not contain some component of the difference between the initial conditions of the two experiments. Third, it should be possible for HYPGENE to generate hypotheses by reasoning forward from the difference between initial conditions; this type of reasoning may be more efficient for some problems than is the backward reasoning that HYPGENE now employs. Although these techniques are similar to those developed by other researchers, Chapter 5 analyzed the techniques in finer detail and in the context of more complex problems than previous researchers have analyzed their techniques.

In Chapter 6 we examined the implementation of HYPGENE in detail to see how HYPGENE's design goals are represented, how it expands new search states, and how it incrementally recomputes the new predicted outcome of an experiment under a given hypothesis. I hope that these implementation details will aid future workers in constructing similar programs. Chapter 6 also compared my hypothesis-formation methods with those developed by other researchers.

Although I developed HYPGENE to solve hypothesis-formation problems in molecular biology, I believe that both the design framework and the specific methods discussed in Chapters 5 and 6 will prove to be applicable to other domains. The methods discussed in Chapter 5 refer to experiments whose initial conditions and outcomes can be expressed as lists of predicate-calculus assertions. They also refer to theories that can be expressed as sets of processes with predicate-calculus preconditions, and whose effects can be formulated as assertion add lists. The design operators described in Chapter 5 manipulate the assertions in experiment descriptions, and the conditions and assertions in process descriptions. No design operators are specific to the domain of molecular biology (with the exception of the quantitative-hypothesis design operators, which incorporate assumptions about my qualitative chemistry). Thus, any domain that we can model using the GENSM framework should be a candidate domain for

HYPGENE, subject to the limitations in Section 2.3.1.

HYPGENE's operators are not only general, they are also complete — in two senses. They are syntactically complete because they are capable of making all possible changes to the GENSIM representation language. For example, the process-design operators can create any process that can be expressed in GENSIM's process-description language. The operators are semantically complete in that the hypotheses that they design encompass all types of causal change in this domain. For example, every causal mechanism that could increase the quantity of an object in a experiment is encoded as an operator. Despite the operator's completeness, I can imagine two reasons that we might want to supplement them. The first is that HYPGENE might be more efficient if it contained macro operators that combined the existing operators in particularly useful ways (see Section 5.8.3 for more details). The second reason is that we might wish to alter the general causal framework in this domain. For example, if we removed GENSIM's assumption that processes never remove objects from a simulation (see Section ??), we would be modifying our conception of how change occurs in this domain, which would require that HYPGENE contain additional operators.

8.4 Future Work

In the short term, my thesis work should be extended in several ways. The device-modeling frameworks used by models 2 and 3 should be combined to produce a system that reasons about both object structures and object concentrations, and that predicts both what chemical reactions occur, and at what rates. The system should also reason about temporal and spatial aspects of devices.

In addition, the dissertation described several methods that have not been implemented, but should be. The process-design operators and class KB design operators should be implemented, as should the two unimplemented techniques for using reference experiments: consistency

checking and forward reasoning.

The historical study in Chapter 4 contained an army of examples of scientific reasoning that must be conquered by implemented programs. Some of these examples involve forms of reasoning that I have identified, such as the design of new processes and forward reasoning from reference experiments. Other examples have no doubt devised treacherous tactics that will baffle future graduate students.

In the longer term, HYPGENE and its descendants should be applied to hypothesis-formation problems in other biological systems, and in other domains. The most convincing demonstrations of hypothesis-formation programs will come (and have come) when these programs solve unresolved scientific problems.

In the future, it will be easier to apply hypothesis-formation programs to unsolved scientific problems, and it will be more important that we do so. The reason is that scientists are now entrusting more and more of their data and knowledge to computers. This situation makes scientific data more accessible to AI researchers and to their hypothesis-formation programs. As scientists accumulate more information, they will use it to address increasingly difficult problems. Hypothesis-formation programs will mature into invaluable assistants that allow scientists to design new theories from these vast amounts of knowledge and data.

Appendix A

Process Knowledge Base

This appendix shows the taxonomic structure of the process knowledge base, and lists the definitions of several processes. Figure A.1 shows all processes in the process knowledge base.

The remainder of this appendix lists the definitions of the following processes:

- Trp-ApoRepressor.Binds.Trp
- Anthranilate-Synthetase.Catalysis
- Trp-Repressor.Binds.Operator
- Transcription.Initiation
- Transcription.Elongation
- Transcription.Termination
- MutableSite.Binds.MutableSite
- Mutation.Check1

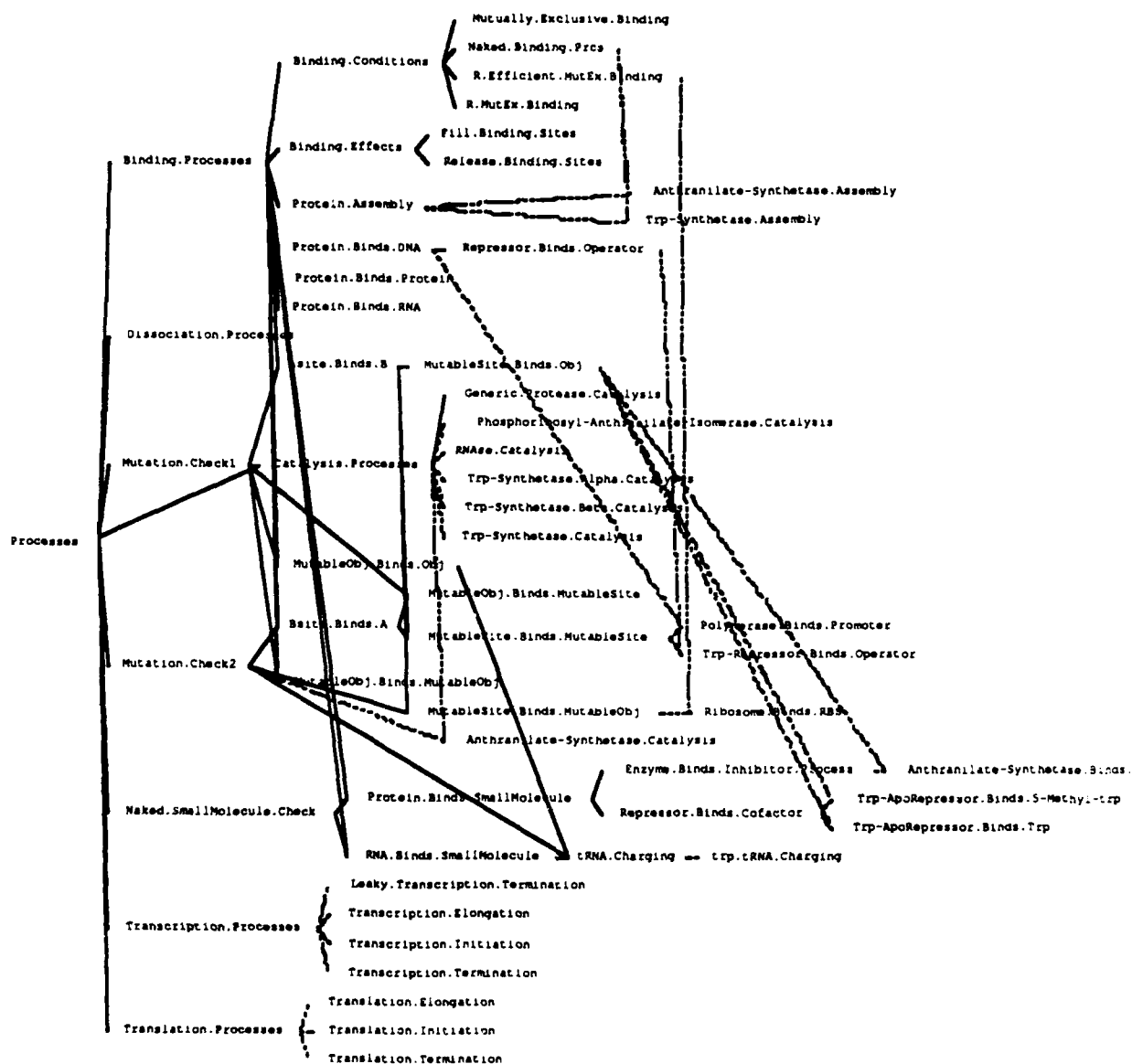


Figure A.1: The process KB. The solid lines in this figure indicate the class-subclass relationship, with more specialized processes to the right. The dashed lines indicate the class-member relationship.

Process Trp-ApoRepressor.Binds.Trp

```

Parameter.Object.Classes: Trp-ApoRepressor trp
Parameter.Objects:      $A $B
Bindings.M:             ($Complex.Class Trp-Repressor)
Bindings.A:             ($Aobj $A)
                       ($Bobj $B)
                       ($mcheck1 $Asite)
                       (* $Aobj <- first object)
                       (* $Bobj <- second object)
                       (* mcheck1 <- Mutable object)

Preconditions.M:
Preconditions.A:        (* * Check that Aobj contains an active site which interacts
                        with objects of Bobj's type.)
                        [EXISTS $Asite
                        (AND (IS.PART.R $Asite $Aobj)
                        (OBJECT.EXISTS $Asite 'Active.Sites)
                        (EXISTS $site.interaction.class
                        (AND (MEMB $site.interaction.class
                        (W/GET.VALUES
                        $Asite
                        'Potential.Interacting.Objects))
                        (OBJECT.EXISTS $Bobj
                        $site.interaction.class]
                        (* * Check that Asite isn't occupied.)
                        [NOT (EXISTS $obj
                        (AND (MEMB $obj (W/GET.VALUE
                        $Asite
                        'Object.Interacting.With.Site))
                        (OBJECT.EXISTS $obj
                        (W/GET.VALUE
                        $Asite
                        'Potential.Interacting.Objects]
                        (* * Mutation check 1)
                        [NOT (EXISTS $mutation (AND (IS.PART $mutation $mcheck1)
                        (OBJECT.EXISTS $mutation 'Mutations)
                        (MEMB $Current.Process
                        (W/GET.VALUES
                        $mutation
                        'Processes.Disabled]
                        [NOT (EXISTS $obj (IS.PART $B $obj)
                        (OBJECT.EXISTS $obj 'Physical.Entities]

Efficiency.Preconditions.A: ...
Effects.M:              (BINDV $bA (COPY.STRUCTURE $A))
                       (BINDV $bB (COPY.STRUCTURE $B))
                       (BINDV $Complex (CREATE.COMPLEX $Complex.Class (LIST $bA $bB)
                       'RBOUND))

Effects.A:              (* * Record that $Asite is interacting with $Bobj)
                       (W/PUT.VALUE (OBJECT.COPIED.TO $Asite)
                       'Object.Interacting.With.Site
                       (OBJECT.COPIED.TO $Bobj))

```

Process Anthranilate-Synthetase.Catalysis

```

Parameter.Object.Classes: Anthranilate-Synthetase Chorismate
Parameter.Objects:      $Enzyme $InSubstrate
Bindings.M:             ($OutSubstrate.Class PRanthranilate)
Bindings.A:             ($Enzyme.Container $Enzyme)
                        ($mcheck2 $inhibition.site)
                        (* mcheck2 <- mutable object)
                        ($mcheck1 $Enzyme)
                        (* mcheck1 <- Mutable object)

Preconditions.M:        ...
Preconditions.A:        (* * Find Anthranilate-Synthetase inhibition site.)
                        [EXISTS $inhibition.site
                          (AND (IS.PART.R $inhibition.site $Enzyme)
                                (OBJECT.EXISTS $inhibition.site
                                  'Active.Sites)

                        (* * Site cannot be occupied.)
                        [NOT
                          (EXISTS $class
                            (EXISTS $obj
                              (AND (MEMB $class
                                    (W/GET.VALUES
                                      $inhibition.site
                                      'Potential.Interacting.Objects))
                                (OBJECT.EXISTS $obj $class)
                                (MEMB $obj (W/GET.VALUES
                                      $inhibition.site
                                      'Object.Interacting.With.Site]

                        (* * Mutation check 2)
                        [NOT (EXISTS $mutation
                              (AND (IS.PART $mutation $mcheck2)
                                    (OBJECT.EXISTS $mutation 'Mutations)
                                    (MEMB $Current.Process
                                      (W/GET.VALUES
                                        $mutation
                                        'Processes.Disabled]

                        (* * Mutation check 1)
                        [NOT (EXISTS $mutation
                              (AND (IS.PART $mutation $mcheck1)
                                    (OBJECT.EXISTS $mutation 'Mutations)
                                    (MEMB $Current.Process
                                      (W/GET.VALUES
                                        $mutation
                                        'Processes.Disabled]

Efficiency.Preconditions.A: ...
Effects.M:               (BINDV $OutSubstrate (CREATE.OBJECT ' PRanthranilate)
Effects.A:               ...

```


Process Trp-Repressor.Binds.Operator

Parameter.Object.Classes:

Trp-Repressor Trp.Operator

Parameter.Objects:

\$A \$B

Bindings.M:

(\$Complex.Class RepOp.Complexes)

Bindings.A:

(\$Aobj \$A)

(\$Bobj \$B)

(\$mcheck2 \$Bsite)

(* mcheck2 <- mutable object)

(\$mcheck1 \$Asite)

(* \$Aobj <- first object)

(* \$Bobj <- second object)

(* mcheck1 <- Mutable object)

Preconditions.M:

...

Preconditions.A:

(* * Check that Bobj contains an active site that interacts
with objects of Aobj's type.)

[EXISTS \$Bsite

(AND (IS.PART.R \$Bsite \$Bobj)

(OBJECT.EXISTS \$Bsite 'Active.Sites)

(EXISTS \$site.interaction.class

(AND (MEMB \$site.interaction.class

(W/GET.VALUES

\$Bsite

'Potential.Interacting.Objects))

(OBJECT.EXISTS \$Aobj

\$site.interaction.class]

(* * Check that Bsite is not occupied.)

[NOT

(EXISTS \$obj

(AND (MEMB \$obj (W/GET.VALUES \$Bsite

'Object.Interacting.With.Site))

(OBJECT.EXISTS \$obj

(W/GET.VALUE

\$Bsite

'Potential.Interacting.Objects]

(* * Mutation check 2)

[NOT (EXISTS \$mutation

(AND (IS.PART \$mutation \$mcheck2)

(OBJECT.EXISTS \$mutation 'Mutations)

(MEMB \$Current.Process

(W/GET.VALUES

\$mutation

'Processes.Disabled]

(* * Check that Aobj contains an active site which interacts
with objects of Bobj's type.)

[EXISTS \$Asite

(AND (IS.PART.R \$Asite \$Aobj)

(OBJECT.EXISTS \$Asite 'Active.Sites)

(EXISTS \$site.interaction.class

(AND (MEMB \$site.interaction.class

(W/GET.VALUES \$Asite

'Potential.Interacting.Objects))

(OBJECT.EXISTS \$Bobj

\$site.interaction.class]

(* * Check that Asite isn't occupied.)

[NOT (EXISTS \$obj

(AND (MEMB \$obj

```

(W/GET.VALUES $Asite
  'Object.Interacting.With.Site))
(OBJECT.EXISTS $obj
  (W/GET.VALUE $Asite
    'Potential.Interacting.Objects]

(* * Mutation check 1)
[NOT (EXISTS $mutation
  (AND (IS.PART $mutation $mcheck1)
    (OBJECT.EXISTS $mutation 'Mutations)
    (MEMB $Current.Process
      (W/GET.VALUES
        $mutation
        'Processes.Disabled]

(* * Do not allow binding if Polymerase is bound to the promoter which
this operator controls. Determine if a binding site exists, which is
within a promoter, where that promoter is controlled by the current
operator, and the binding site is bound to Polymerase.
(Historically this isn't completely accurate, i.e. this behavior was
determined in '78))
[NOT
  (EXISTS $pro
    (AND (OBJECT.EXISTS $pro 'Promoters)
      (MEMB $pro (W/GET.VALUES $B 'Promoters.Controlled))
      (EXISTS $BindingSite
        [AND (OBJECT.EXISTS $BindingSite 'Active.Sites)
          (IS.PART $BindingSite $pro)
          (EXISTS $obj
            (AND (MEMB $obj
              (W/GET.VALUES
                $BindingSite
                'Object.Interacting.With.Sit.
              (OBJECT.EXISTS $obj
                'RNA-Polymerase]

NIL]

Efficiency.Preconditions.A:
  (NOT (HAS.COMPONENT.R (ROOT.CONTAINER $A)
    'RNA-Polymerase))

Effects.M:
  (BINDV $ba (COPY.STRUCTURE $A))
  (BINDV $bb (COPY.STRUCTURE $B))
  (BINDV $Complex (CREATE.COMPLEX $Complex.Class
    (LIST $ba $bb)
    'RBOUND))

Effects.A:
  (* * Record that $Aobj is interacting with $Bsite)
  (W/PUT.VALUE (OBJECT.COPIED.TO $Bsite)
    'Object.Interacting.With.Site
    (OBJECT.COPIED.TO $Aobj))
  (* * Record that $Asite is interacting with $Bobj)
  (W/PUT.VALUE (OBJECT.COPIED.TO $Asite)
    'Object.Interacting.With.Site
    (OBJECT.COPIED.TO $Bobj))
  (PUT.VALU' (W/GET.VALUE (OBJECT.COPIED.TO $B)
    'Promoters.Controlled)
    'Receptive.To.Polymerase
    'NO)

```

Process Transcription.Initiation

Parameter.Object.Classes:

XCInit.Complexes

Parameter.Objects: \$XCInit.Complex

Bindings.M: ...

Bindings.A: ...

```
Preconditions.M: (EXISTS $Promoter (AND (OBJECT.EXISTS $Promoter
                                     'Promoters)
                                     (IS.PART.R $Promoter
                                      $XCInit.Complex)))
                 (EXISTS $Operon (AND (OBJECT.EXISTS $Operon 'Operons)
                                     (IS.PART $Operon $XCInit.Complex)))
```

Preconditions.A: ...

Efficiency.Preconditions.A: ...

```
Effects.M: (BINDV $Rnap (HAS.COMPONENT.R $XCInit.Complex
                                     'RNA-Polymerase))
            (BINDV $NEW.Rnap (COPY.SUBSTRUCTURE $Rnap))
            (BINDV $NEW.Operon (COPY.SUBSTRUCTURE $Operon))
            (BINDV $mRNA (CREATE.OBJECT 'Messenger.RNAs))
            (BINDV $XCElong.Complex
              (CREATE.COMPLEX 'XCElong.Complexes
                              (LIST $mRNA $NEW.Operon $NEW.Rnap)
                              'IRBOUND))
            (BINDV $DNA.Segment (W/GET.VALUE $Promoter
                                     '3Prime.Segment))
            [BINDV $mRNA.Segment
              (CREATE.OBJECT (W/GET.VALUE $DNA.Segment
                                     'RNA.Segment.Produced]
              (W/PUT.VALUE $mRNA.Segment 'Source.DNA.Segment
                              (OBJECT.COPIED.TO $DNA.Segment $NEW.Operon))
              (W/PUT.VALUE $mRNA.Segment 'Source.DNA.Segment
                              (OBJECT.COPIED.TO $DNA.Segment $NEW.Operon))
              (ADD.PART $mRNA $mRNA.Segment 'IRBOUND)
              (W/PUT.VALUE $mRNA 'First.Segment
                              $mRNA.Segment)
              (W/PUT.VALUE $mRNA 'Transcribed.Operon
                              $NEW.Operon)
```

Effects.A: ...

Process Transcription.Elongation

Parameter.Object.Classes:

XCElong.Complexes

Parameter.Objects: \$XCElong.Complex

Bindings.M: ...

Bindings.A: ...

```

Preconditions.M: (EXISTS $mRNA (AND (OBJECT.EXISTS $mRNA 'Messenger.RNAs)
                                (IS.PART $mRNA $XCElong.Complex)))
                (EXISTS $RNAp (AND (OBJECT.EXISTS $RNAp 'RNA-Polymerase)
                                (IS.PART $RNAp $XCElong.Complex)))
                (EXISTS $Operon (AND (OBJECT.EXISTS $Operon 'Operons)
                                (IS.PART $Operon $XCElong.Complex)))
                (NOT (OBJECT.EXISTS
                    (W/GET.VALUE
                        (W/GET.VALUE (FOLLOW.SLOT.POINTERS
                            (W/GET.VALUE
                                $mRNA
                                'First.Segment)
                                '3Prime.Segment)
                                'Source.DNA.Segment)
                            '3Prime.Segment)
                        'Terminators)))

```

Preconditions.A: ...

Efficiency.Preconditions.A: ...

```

Effects.M: (BINDV $New.XCElong.Complex (COPY.STRUCTURE
                                         $XCElong.Complex))
           (BINDV $NEW.mRNA (HAS.COMPONENT.R $New.XCElong.Complex
                                         'Messenger.RNAs))
           (BINDV $Last.mRNA.Segment
               (FOLLOW.SLOT.POINTERS (W/GET.VALUE
                                       $mRNA
                                       'First.Segment))
               '3Prime.Segment))
           (BINDV $DNA.Segment
               (W/GET.VALUE (W/GET.VALUE $Last.mRNA.Segment
                                       'Source.DNA.Segment)
                           '3Prime.Segment))
           [BINDV $New.mRNA.Segment
             (CREATE.OBJECT
                 (W/GET.VALUE $DNA.Segment
                     'RNA.Segment.Produced]
           (ADD.PART $NEW.mRNA $New.mRNA.Segment 'IRBOUND)
           (W/PUT.VALUE $New.mRNA.Segment
               'Source.DNA.Segment
               (OBJECT.COPIED.TO $DNA.Segment
                               $New.XCElong.Complex))
           (W/PUT.VALUE (OBJECT.COPIED.TO $Last.mRNA.Segment
                                       $New.XCElong.Complex)
               '3Prime.Segment
               $New.mRNA.Segment)

```

Effects.A: ...

Process Transcription.Termination

Parameter.Object.Classes:

XCElong.Complexes

Parameter.Objects: \$XCElong.Complex

Bindings.M: ...

Bindings.A: ...

```

Preconditions.M: (EXISTS $mRNA (AND (OBJECT.EXISTS $mRNA 'Messenger.RNAs)
                                (IS.PART $mRNA $XCElong.Complex)))
                (EXISTS $RNaP (AND (OBJECT.EXISTS $RNaP 'RNA-Polymerase)
                                (IS.PART $RNaP $XCElong.Complex)))
                (EXISTS $Operon (AND (OBJECT.EXISTS $Operon 'Operons)
                                (IS.PART $Operon $XCElong.Complex)))
                (OBJECT.EXISTS
                  (W/GET.VALUE
                    (W/GET.VALUE (FOLLOW.SLOT.POINTERS
                                (W/GET.VALUE $mRNA
                                  'First.Segment)
                                  '3Prime.Segment)
                                'Source.DNA.Segment)
                              '3Prime.Segment)
                  'Terminators)

```

Preconditions.A: ...

Efficiency.Preconditions.A: ...

Effects.M: (BINDV \$Free.mRNA (COPY.SUBSTRUCTURE \$mRNA))

Effects.A: ...

Process MutableSite.Binds.MutableSite

```

Parameter.Object.Classes: Physical.Entities Physical.Entities
Parameter.Objects:      $A $B
Bindings.M:             ($Complex.Class Molecular.Complexes)
Bindings.A:             ($Aobj $A)
                       ($Bobj $B)
                       ($mcheck2 $Bsite)
                       (* mcheck2 <- mutable object)
                       ($mcheck1 $Asite)
                       (* $Aobj <- first object)
                       (* $Bobj <- second object)
                       (* mcheck1 <- Mutable object)

Preconditions.M:        ...
Preconditions.A:        (* * Check that Bobj contains an active site that interacts
                        with objects of Aobj's type.)
[EXISTS $Bsite
  (AND (IS.PART.R $Bsite $Bobj)
    (OBJECT.EXISTS $Bsite 'Active.Sites)
    (EXISTS $site.interaction.class
      (AND (MEMB $site.interaction.class
        (W/GET.VALUES
          $Bsite
          'Potential.Interacting.Objects))
        (OBJECT.EXISTS $Aobj
          $site.interaction.class]
    (* * Check that Bsite is not occupied.)
  [NOT (EXISTS $obj
    (AND (MEMB $obj
      (W/GET.VALUES
        $Bsite
        'Object.Interacting.With.Site))
      (OBJECT.EXISTS $obj
        (W/GET.VALUE $Bsite
          'Potential.Interacting.Objects]
    (* * Mutation check 2)
  [NOT (EXISTS $mutation
    (AND (IS.PART $mutation $mcheck2)
      (OBJECT.EXISTS $mutation
        'Mutations)
      (MEMB $Current.Process
        (W/GET.VALUES
          $mutation
          'Processes.Disabled]
    (* * Check that Aobj contains an active site which
      interacts with objects of Bobj's type.)
  [EXISTS $Asite
    (AND (IS.PART.R $Asite $Aobj)
      (OBJECT.EXISTS $Asite 'Active.Sites)
      (EXISTS $site.interaction.class
        (AND (MEMB $site.interaction.class
          (W/GET.VALUES $Asite
            'Potential.Interacting.Objects))
          (OBJECT.EXISTS $Bobj
            $site.interaction.class]
    (* * Check that Asite isn't occupied.)
  [NOT (EXISTS $obj
    (AND (MEMB $obj (W/GET.VALUES $Asite

```


Process Mutation.Check1

```

Parameter.Object.Classes: ...
Parameter.Objects: ...
Bindings.M: ...
Bindings.A: (* mcheck1 <- Mutable object)
Preconditions.M: ...
Preconditions.A: (* * Mutation check 1)
                  [NOT (EXISTS $mutation
                        (AND (IS.PART $mutation $mcheck1)
                            (OBJECT.EXISTS $mutation
                                'Mutations)
                            (MEMB $Current.Process
                                (W/GET.VALUES
                                    $mutation
                                    'Processes.Disabled])

Efficiency.Preconditions.A: ...
Effects.M: ...
Effects.A: ...

```


Appendix B

Class Knowledge Base

This appendix provides an overview of the GENSIM class knowledge base (CKB). Figures B.1, B.2, and B.3 show three different portions of the KB. These figures can be pieced together vertically to diagram the entire KB. General classes of objects are specialized to subclasses to their right.

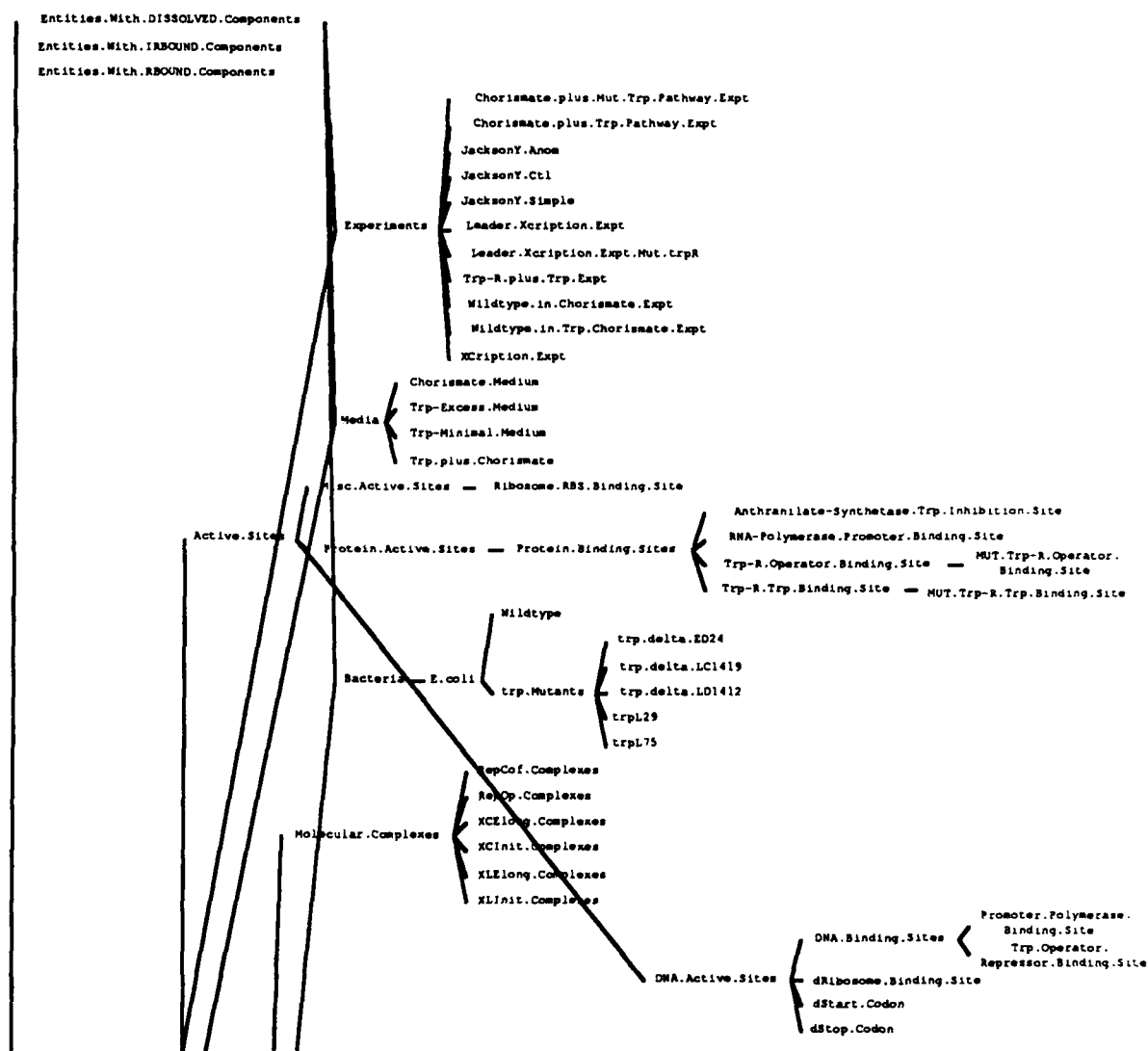


Figure B.1: The top third of the class KB.

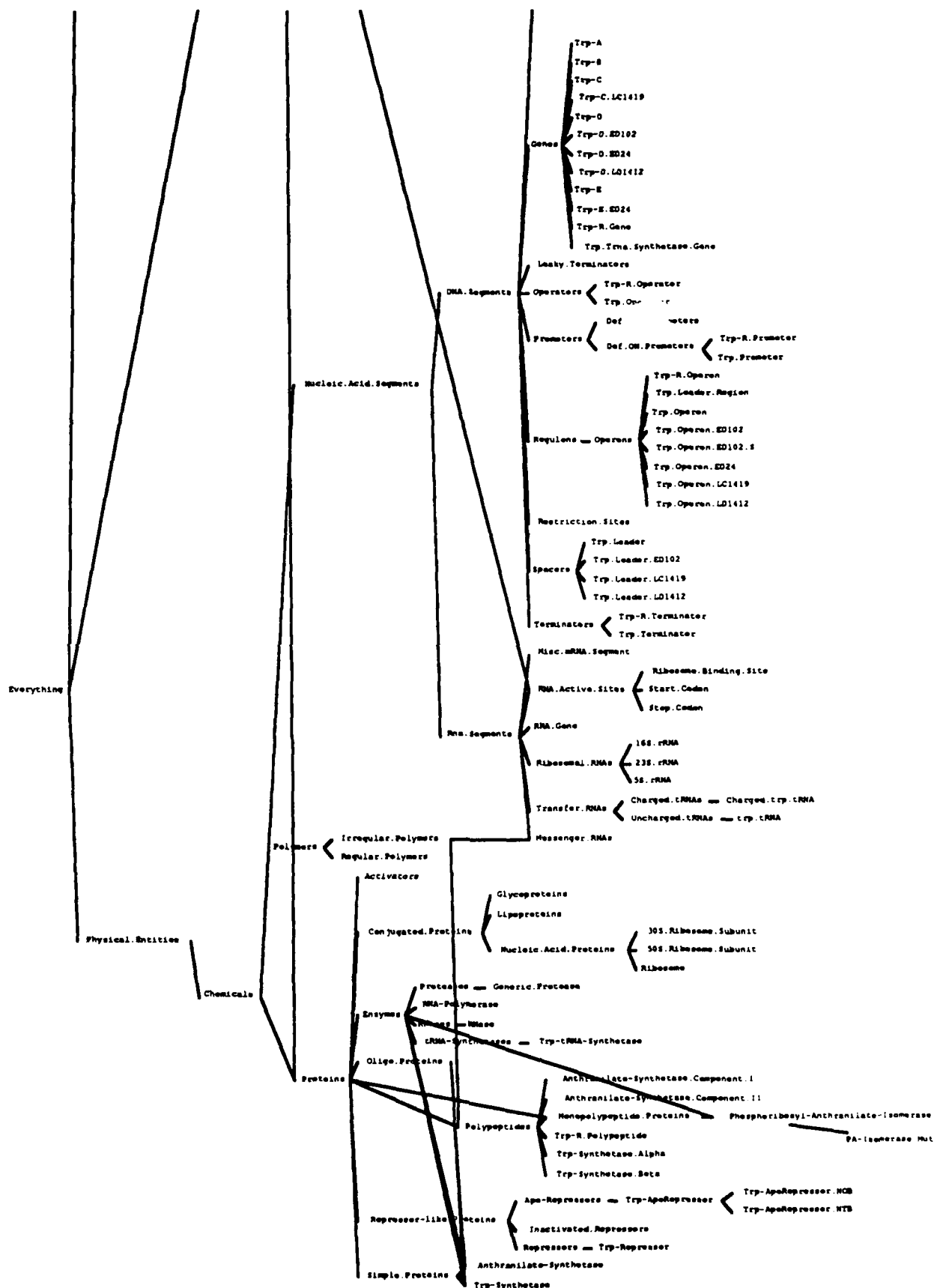


Figure B.2: The middle third of the class KB.

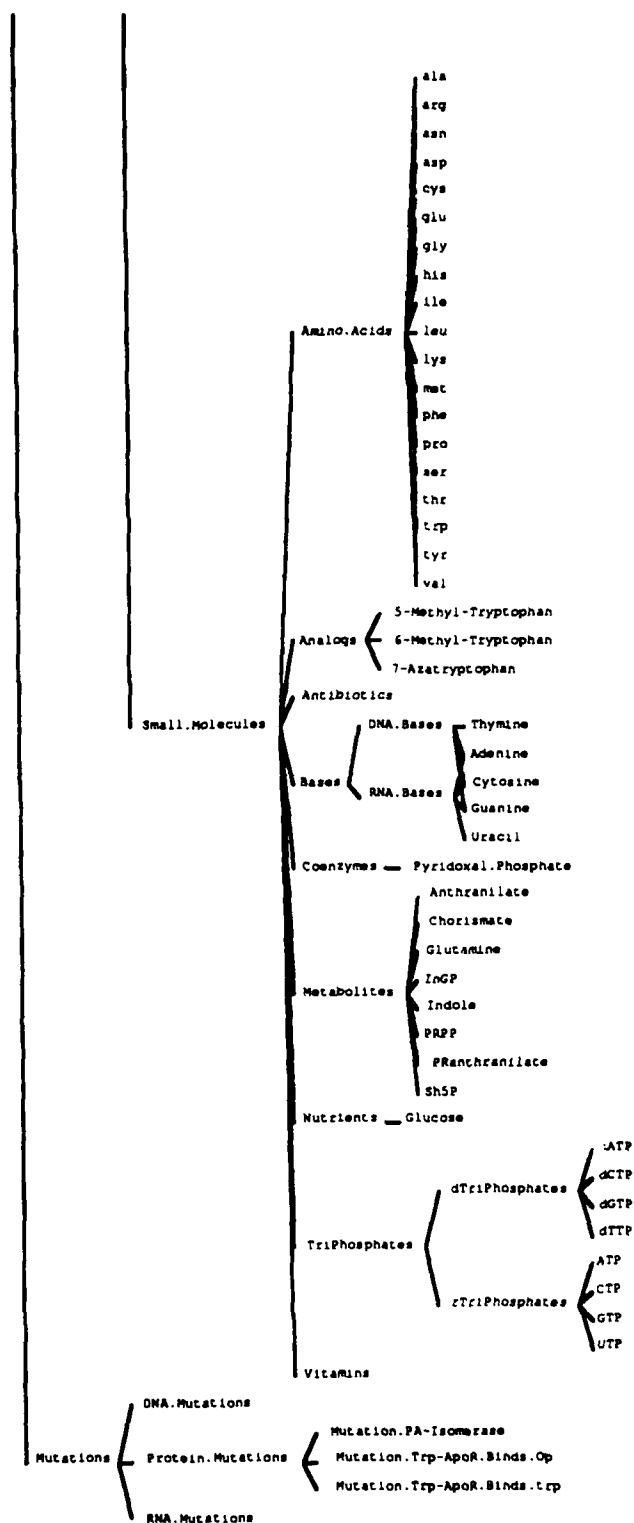


Figure B.3: The bottom third of the class KB.

Appendix C

Attenuation Paper Summaries

For each unit of research described in Chapter 4, this appendix provides

- A summary of what phenomenon was studied
- A summary of what new knowledge this unit of research produced
- A description of what modes of exploration were employed in this unit
- A description of what types of syntactic modifications were made to theories of the trp operon

Table C.1 defines the mnemonics for the different types of theory modifications.

C.1 State 1

[Hiraga69]

- Studied: Trp operator and repressor mutants
- Found: Characterized their behavior. Estimated size of leader region.
- Modes: CO
- Changes: $\Delta B+$, NO, EX:CM

[ItoHY69]

- Studied: trp-tRNA-synthetase mutants
- Found: Elevated levels of trp operon expression
- Modes: CO
- Changes: $\Delta B-$, EX:CM

Mnemonic	Meaning
EX	Extend domain of applicability
EX:CM	Characterize Mutant
RP	Refine parameter value
NO:I	Instantiate new object from old class
NO:R	Refine existing object
NO:P	Instantiate new object from new class
MPR	Modify Process
CP	Create Process
$\Delta B+$	Increase credibility
$\Delta B-$	Decrease credibility

Table C.1: Mnemonics used to describe theory modifications.

[Imamoto70]

- Studied: Time course of the onset of repression
- Anomaly: mRNA production began sooner than expected
- Modes: CO
- Changes: $\Delta B-$, RP

[YanofskyH72]

- Studied: correspondences between structures of trp-A protein and gene
- Modes: CO
- Changes: $\Delta B+$, NO

[ForchhammerJY72]

- Studied: Trp-mRNA degradation
- Found: Quantified rate of degradation; showed degradation proceed
- Modes: CO
- Changes: RP, MPR(Transcription), NoLink(DegradationRate,mRNA-Sequence)

[CohenYY73]

- Found: Sequence of 5' end of trp-mRNA
- Modes: CO
- Changes: NO:R

[JacksonY72]

- Studied: P_2 promoter within trpD gene
- Found: Mapped its position and established its efficiency at 5
- Modes: CO
- Changes: RP, NO:I

[BakerY72]

- Studied: Trp operon transcription and translation rates
- Found: Quantified rates of these processes

- Anomaly: A higher rate of transcription was observed right after wildtype cells were placed in a trp-free medium than for trpR- cells.
- Modes: CO
- Changes: $\Delta B-$, RP, RP

[RoseY72]

- Studied: Trp-mRNA synthesis rates in different media
- Anomaly: An unknown mechanism appeared to increase operon expression in richer media, independent of trp concentration
- Modes: CO
- Changes: $\Delta B-$, RP, CP(Medium-Richness influences Transcription-Rate)

C.2 State 2

[JacksonY73]

- Studied: Leader region deletion mutants
- Measured: mRNA degradation, operon copy number, location of deletion, mRNA synthesis
- Anomaly: Deletions elevated maximal synthesis levels
- Hypothesis: Regulatory region exists in leader region
- Modes: CO, TG, DI
- Changes: $\Delta B-$, EX:CM, RP(3), CP(Attenuator influences Transcription-Rate)

[Kasai74]

- Studied: Deletion mutants in start of his operon
- Anomaly: Elevated operon expression. Mutant tRNA^{his} makes his operon constitutive
- Hypothesis: tRNA^{his} regulates binding of a protein factor during transcription initiation which accompanies RNA Polymerase and prevents attenuation.
- Modes: CO, TG
- Changes: $\Delta B-$, EX:CM, NO:P, CP(Protein factor binds polymerase, rate proportional to un-factored polymerase)

C.3 State 3

[ArtzB75]

- Studied: his operon mutants
- Found: in vitro his operon transcription requires presence of a translation system; also, if DNA concentration is steadily increased, gene expression soon reaches a maximum, suggesting a positive factor is being used up
- Modes: CO
- Changes: $\Delta B+$, EX:CM, CP(Translation influences Transcription-rate), CP([DNA] influences Transcription-rate)

[PlattSY76]

- Studied: Ribosome binding sites in the 5' end of the trp operon
- Found: One at start of trpE, one in leader region

- Anomaly: Ribosome binding site in leader region
- Modes: FF
- Changes: $\Delta B-$, NO:I

[PlattY75]

- Studied: Ribosome binding sites in all trp operon genes
- Found: translation-termination sites and ribosome binding sites overlap
- Modes: FF
- Changes: $\Delta B-$, NO:I

[SquiresLY75]

- Studied: in vitro interactions of RNA polymerase, trp repressor, promoter
- Found: Trp repressor inhibits transcription by preventing binding of RNA polymerase; repressor cannot prevent transcription if polymerase is pre-incubated with promoter. Pre-incubated polymerase will prevent repressor from binding to the operator
- Modes: FF
- Changes: MPR (Several binding processes)

[LeeSSY76]

- Studied: in vitro transcription termination at the attenuator
- Found: Growth of nascent mRNA could be observed; it never elongated past the attenuator
- Modes: CO
- Changes: $\Delta B+$

[BertrandY76]

- Found: Attenuation is regulated by trp concentration
- Modes: ???
- Changes: MPR

[BertrandSY76]

- Studied: Re-analysis of leader-deletion mutants
- Found: Detailed evidence of premature transcription termination at the leader region: trp promoter efficiency is unaffected; deletions are operator-distal; leader-region deletions are cis-dominant (thus no diffuseable element is destroyed).
- Modes: CO, DI
- Changes: $\Delta B+$, EX:CM, RP, NO:R

[SquiresLBSBY76]

- Found: Sequence of leader region trp-mRNA
- Modes: FF
- Changes: NO:R

[KornY76]

- Studied: interactions of rho with polarity and attenuation
- Found: Polarity suppressed in rho mutants; reduced attenuation also
- Modes: FF
- Changes: EX:CM, CP(rho influences Polarity), CP(rho influences Attenuation-rate)

C.4 State 4

[MorseM76]

- Studied: Involvement of tRNA^{trp} in trp operon regulation
- Found: Attenuation is sensitive to charging of tRNA^{trp}
- Modes: FF
- Changes: $\Delta B+$, CP(charged-tRNA^{trp} influences Attenuation-rate)

[YanofskyS77]

- Found: More evidence that tRNA^{trp} is involved in attenuation
- Modes: CO, DI
- Changes: $\Delta B+$

[BertrandKLY77]

- Studied: 3' terminus in leader transcript
- Found: It contains an OH group, indicative of transcription termination as opposed to digestion
- Modes: CO
- Changes: $\Delta B+$, NO:R

[LeeY77]

- Studied: *E. coli* and *S. typhimurium* trp operons
- Found: RNase digestion revealed the leader hairpin secondary structures. The @i(*S. typhimurium*) hairpin was predicted to be less stable, which correlates with its higher transcription read-through rate. Mutating it to make it less stable decreased attenuation further. Sequence analysis showed two mutually exclusive secondary structure loops.
- Hypothesis: The alternative secondary structures form a switch
- Modes: FF, TG, CO
- Changes: $\Delta B+$, EX:CM, NO:R, CP(secondary structures cause termin

[MiozzariY78a]

- Studied: trp operons of *E. coli* and *S. marcescens*
- Found: Leader region hairpins are conserved.
- Modes: CO
- Changes: $\Delta B+$, EX, NO:R

[MiozzariY78b]

- Studied: Translation of the leader peptide as a gene fusion
- Found: Leader peptide ribosome binding site can direct translation
- Modes: CO
- Changes: $\Delta B+$

[ZurawskiBKY78]

- Studied: *E. coli* phenylalanine operon
- Found: Leader region ribosome binding site, translation start codon, in-phase translation stop codon, 3'-OH group on mRNA, mRNA hairpin structures, 7 phenylalanine codons.
- Modes: CO
- Changes: $\Delta B+$, EX, NO:R

[OxenderZY79]

- Studied: Leader region mRNA secondary structures

- Found: Formation of alternative structures is correlated with presence and absence of leader region transcription termination
- Modes: CO, TG, CO
- Changes: $\Delta B+$, EX:CM

[BennettY78]

- Studied: Location of trp operator
- Found: Its location was precisely determined using two different assays
- Modes: FF
- Changes: NO:R

[BrownBLSY78]

- Studied: Location of trp promoter
- Found: Its location was precisely determined using two different assays
- Modes: FF
- Changes: NO:R

[BennettSBSY78]

- Studied: DNA sequence of promoter-operator region
- Modes: FF
- Changes: NO:R

C.5 State 5

[StroynowskiY82]

- Studied: Impact of controlled hairpin deletions on attenuation
- Found: Deleting specific regions affected attenuation: sometimes in the expected way, sometimes subtle explanations were constructed
- Modes: CO
- Changes: $\Delta B+$, EX:CM

[DasCY82]

- Studied: Regulation of trp operon in vitro
- Found: The trp operon can be regulated in vitro by attenuation
- Modes: FF
- Changes: EX

[ZurawskiY80]

- Studied: Trp operon mutants with high expression rates; these strains were further mutated and the experimenters selected for resulting mutants with low expression
- Found: Most such low-expression mutants had a destabilized 3:4 stem
- Modes: CO
- Changes: $\Delta B+$, EX:CM

[Platt81]

- Studied: General aspects of transcription termination
- Modes: Review
- Changes: Review

[NicholsVY81]

- Studied: Sequence of *trpE* gene
- Found: It contains no *trp* residues
- Modes: FF
- Changes: NO:R

[Yanofsky83]

- Studied: Codon frequencies in *trpE* in *S. typhimurium* and *E. coli*
- Modes:
- Changes:

[KelleyY82]

- Studied: Regulation of *trp* repressor production
- Hypothesized: An explanation of why it regulates its own production
- Modes: TG, CO
- Changes: RP

[Das82]

- Studied: in vitro expression of leader peptide
- Found: Leader peptide can be expressed in vitro; its synthesis is shut off by a downstream sequence
- Modes: CO, TG
- Changes: $\Delta B+$, NO, CP

[Dekel-GorodetskySE86]

- Studied: in vivo expression of leader peptide
- Found: Can be expressed in vivo
- Modes: CO
- Changes: $\Delta B+$

Bibliography

- [1] R.B. Altman. *Exclusion Methods for the Determination of Protein Structure from Experimental Data*. PhD thesis, Stanford University Medical Information Sciences, 1989.
- [2] S. Artz. Personal Communication, 1986.
- [3] S.W. Artz and J.R. Broach. Histidine regulation in *S. typhimurium*: An activator-attenuator model of gene regulation. *Proceedings of the National Academy of Sciences, USA*, 72(9):3453-3457, 1975.
- [4] R. Baker and C. Yanofsky. Transcription initiation frequency and translation yield for the tryptophan operon of *E. coli*. *Journal of Molecular Biology*, 69:89-102, 1972.
- [5] B.N. Bennett, M.E. Schweingruber, K.D. Brown, C. Squires, and C. Yanofsky. Nucleotide sequence of the promoter-operator region of the tryptophan operon of *E. coli*. *Journal of Molecular Biology*, 121:113-137, 1978.
- [6] G.N. Bennett and C. Yanofsky. Sequence analysis of operator constitutive mutations of the tryptophan operon of *E. coli*. *Journal of Molecular Biology*, 121:179-192, 1978.
- [7] K. Bertrand, K. Korn, F. Lee, T. Platt, C.L. Squires, C. Squires, and C. Yanofsky. New features of the regulation of the tryptophan operon. *Science*, 189:22-26, 1975.
- [8] K. Bertrand, L.J. Korn, F. Lee, and C. Yanofsky. The attenuator of the tryptophan operon of *E. coli* heterogeneous 3'-OH termini in vivo and deletion mapping of functions.

- Journal of Molecular Biology*, 117:227-247, 1977.
- [9] K. Bertrand, C. Squires, and C. Yanofsky. Transcription termination in vivo in the leader region of the tryptophan operon of *E. coli*. *Journal of Molecular Biology*, 103:319-337, 1976.
- [10] K. Bertrand and C. Yanofsky. Regulation of transcription termination in the leader region of the tryptophan operon of *E. coli* involves tryptophan or its metabolic product. *Journal of Molecular Biology*, 103:339-349, 1976.
- [11] R.L. Blum. *Discovery and representation of causal relationships from a large time-oriented clinical database: The RX project*. PhD thesis, Stanford University, 1982. See also *Computers and Biomedical Research* (15), 1982, pp. 164-187.
- [12] D. G. Bobrow and P. J. (eds) Hayes. *AI Journal Special Issue on Qualitative Reasoning about Physical Systems*. North-Holland, 1984.
- [13] K.D. Brown, G.N. Bennett, F. Lee, M.E. Schweingruber, and C. Yanofsky. RNA polymerase interaction at the promoter-operator region of the tryptophan operon of *E. coli* and *S. typhimurium*. *Journal of Molecular Biology*, 121:153-177, 1978.
- [14] B.G. Buchanan. *Logics of Scientific Discovery*. PhD thesis, Michigan State University Department of Philosophy, 1966.
- [15] B.G. Buchanan and T.M. Mitchell. Model-directed learning of production rules. In *Pattern-Directed Inference Systems*, pages 297-312. Academic Press, 1978.
- [16] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333-377, 1987.
- [17] P.T. Cohen, M. Yaniv, and C. Yanofsky. Nucleotide sequences from messenger RNA transcribed from the operator-proximal portion of the tryptophan operon of *E. coli*. *Journal of Molecular Biology*, 74:163-177, 1973.

- [18] L. Darden and R. Rada. Hypothesis formation using part-whole interrelations. In *Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science, and Philosophy*, pages 341-375. Kluwer, 1988.
- [19] A. Das, I.P. Crawford, and C. Yanofsky. Regulation of tryptophan operon expression by attenuation in cell-free extracts of *E. coli*. *Journal of Biological Chemistry*, 257(15):8795-8798, 1982.
- [20] A. Das, J. Urbanowski, H. Weissbach, J. Nestor, and C. Yanofsky. In vitro synthesis of the tryptophan operon leader peptides of *E. coli*, *S. marcescens*, and *S. typhimurium*. *Proceedings of the National Academy of Sciences, USA*, 80:2879, 1983.
- [21] R. Davis. *Applications of Meta-level Knowledge to the Construction, Maintenance, and Use of Large Knowledge Bases*. PhD thesis, Stanford University Computer Science Department, June 1976. See also Stanford University Computer Science Department reports STAN-CS-76-552 and HPP-76-7.
- [22] R. Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24(1-3):347-410, 1984.
- [23] J. De Kleer. An assumption-based TMS. *Artificial Intelligence*, 28(1):127-162, 1986.
- [24] J. De Kleer and J. S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24(1-3):7-84, 1984.
- [25] L. Dekel-Gorodetsky, R. Schoulaker-Schwarz, and H. Engelberg-Kulka. *E. coli* tryptophan operon directs the in vivo synthesis of a leader peptide. *Journal of Bacteriology*, 165:1046, 1986.
- [26] J. DeKleer and B.C. Williams. Reasoning about multiple faults. In *Proceedings of the 1986 National Conference on Artificial Intelligence*, pages 132-139. Morgan Kaufmann Publishers, 1986.

- [27] T.G. Dietterich. *Constraint Propagation Techniques for Theory-Driven Data Interpretation*. PhD thesis, Stanford University, December 1984. Also STAN-CS-84-1030 and HPP Report No. 84-46.
- [28] T.G. Dietterich and J.S. Bennett. The test incorporation theory of problem-solving (preliminary report). In *Proceedings of the Workshop on Knowledge Compilation*, September 1986.
- [29] T.G. Dietterich and B.G. Buchanan. The role of the critic in learning systems. Technical Report HPP-81-19, Stanford University, 1981. See also Stanford University Computer Science Department report STAN-CS-81-891.
- [30] B.S. Duncan. *Computation of Protein Structures from Experimental Data*. PhD thesis, Stanford University Biophysics Department, 1989.
- [31] B. Falkenhainer. The utility of difference-based reasoning. In *Proceedings of the 1988 National Conference on Artificial Intelligence*, pages 530-535. Morgan Kaufmann Publishers, 1988.
- [32] R.E. Fikes and N.J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189-208, 1971.
- [33] K. Forbus. Qualitative process theory. Technical Report TR-789, Massachusetts Institute of Technology AI Laboratory, 1984.
- [34] K. Forbus. The qualitative process engine. Technical Report UIUCDCS-R-86-1288, University of Illinois Computer Science Department, 1986.
- [35] J. Forchhammer, E. Jackson, and C. Yanofsky. Different half-lives of messenger RNA corresponding to different segments of the tryptophan operon of *E. coli*. *Journal of Molecular Biology*, 71:687-699, 1972.

- [36] P. Friedland. *Knowledge-based Hierarchical Planning in Molecular Genetics*. PhD thesis, Stanford University Computer Science Department, September 1979. Report CS-79-760.
- [37] M.R. Genesereth. The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24(1-3):411-436, 1984.
- [38] A. Goldberg and D. Robson. *Smalltalk-80: The Language and Its Implementation*. Addison-Wesley, 1983.
- [39] Russell Greiner. *Learning by Understanding Analogies*. PhD thesis, Stanford University Computer Science Department, September 1985. See also Stanford University Computer Science Department report CS-85-1071.
- [40] F. Hayes-Roth. Using proofs and refutations to learn from experience. In *Machine Learning*. Tioga, Palo Alto, CA, 1982.
- [41] C.G. Hempel. *Philosophy of Natural Science*. Prentice-Hall, 1966.
- [42] S. Hiraga. Operator mutants of the tryptophan operon in *E. coli*. *Journal of Molecular Biology*, 39:159-179, 1969.
- [43] S. Hiraga. Personal communication. 1984.
- [44] J.Y. Hsu. On the relationship between partial evaluation and explanation-based learning. Technical Report Logic-88-10, Stanford University, 1988.
- [45] F. Imamoto. Immediate cessation of transcription of an operator-proximal segment of the tryptophan operon in *E. coli* following repression of the operon. *Molec. Gen. Genetics*, 106:123-138, 1970.
- [46] F. Imamoto. Personal communication. 1984.
- [47] IntelliCorp. *KEEworlds Reference Manual*, 1986.

- [48] K. Ito, S. Hiraga, and T. Yura. Tryptophanyl transfer RNA synthetase and expression of the tryptophan operon in the *trpS* mutants of *E. coli*. *Genetics*, 61:521-538, March 1969.
- [49] Y. Iwasaki and H.A. Simon. Causality in device behavior. *Artificial Intelligence*, 29:3-32, 1986.
- [50] E.N. Jackson and C. Yanofsky. The region between the operator and first structural gene of the tryptophan operon of *E. coli* may have a regulatory function. *Journal of Molecular Biology*, 76:89-101, 1973.
- [51] F. Jacob and J. Monod. Genetic regulatory mechanisms in the synthesis of proteins. *Journal of Molecular Biology*, 3:318, 1961.
- [52] P.D. Karp and D.C. Wilkins. An analysis of the distinction between deep and shallow expert systems. *International Journal of Expert Systems*, 1989. In press; see also Stanford University Knowledge Systems Laboratory report KSL-89-10.
- [53] T. Kasai. Regulation of the expression of the histidine operon in *S. typhimurium*. *Nature*, 249:523-527, 1974.
- [54] R.L. Kelley and C. Yanofsky. Trp aporepressor production is controlled by autogenous regulation and inefficient translation. *Proceedings of the National Academy of Sciences, USA*, 79:3120-3124, 1982.
- [55] L.J. Korn and C. Yanofsky. Polarity suppressors defective in transcription termination at the attenuator of the tryptophan operon of *E. coli* have altered rho factor. *Journal of Molecular Biology*, 106:231-241, 1976.
- [56] P. Koton. Towards a problem solving system for molecular genetics. Technical Report 338, Massachusetts Institute of Technology Laboratory for Computer Science, 1985.

- [57] B. Kuipers. The limits of qualitative simulation. In *Proceedings of the 1985 International Joint Conference on Artificial Intelligence*, pages 128–136. Morgan Kaufmann Publishers, 1985.
- [58] B.J. Kuipers. Commonsense reasoning about causality: deriving behavior from structure. *Artificial Intelligence*, 24:169–204, 1984.
- [59] D. Kulkarni and H.A. Simon. The processes of scientific discovery: The strategy of experimentation. *Cognitive Science*, pages 139–175, 1988.
- [60] R. Landick, J. Carey, and C. Yanofsky. Translation activates the paused transcription complex and restores transcription of the trp operon leader region. *Proceedings of the National Academy of Sciences, USA*, 82:4663–4667, 1985.
- [61] P. Langley, H.A. Simon, G.L. Bradshaw, and J.M. Zytkow. *Scientific Discovery: Computational Explorations Of The Creative Process*. MIT Press, 1987.
- [62] C.P. Langlotz, L.M. Fagan, S.W. Tu, B.I. Sikic, and E.H. Shortliffe. Combining artificial intelligence and decision analysis for automated therapy planning assistance. Technical Report KSL-86-3, Stanford University Knowledge Systems Laboratory, 1986.
- [63] R. L. Lathrop, T. A. Webster, and T. F. Smith. Ariadne: Pattern-directed inference and hierarchical abstraction in protein structure recognition. *Communications of the Association for Computing Machinery*, 30(11), 1987.
- [64] F. Lee, C.L. Squires, C. Squires, and C. Yanofsky. Termination of transcription in vitro in the *E. coli* tryptophan operon leader region. *Journal of Molecular Biology*, 103:383–393, 1976.
- [65] F. Lee and C. Yanofsky. Transcription termination at the trp operon attenuators of *E. coli* and *S. typhimurium*: RNA secondary structure and regulation of termination. *Proceedings of the National Academy of Sciences, USA*, 74(10):4365–4369, 1977.

- [66] J.A. Lewis and B.N. Ames. Histidine regulation in *S. typhimurium* XI: The percentage of his-tRNA charged in vivo and its relation to the repression of the histidine operon. *Journal of Molecular Biology*, 66:131-142, 1972.
- [67] R. Lindsay, B. G. Buchanan, Feigenbaum E. A., and J. Lederberg. *Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project*. New York: McGraw-Hill, 1980.
- [68] S. Meyers. A simulator for regulatory genetics and its application to bacteriophage lambda. *Nucleic Acids Research*, 12(1):1-9, 1984. Also available as Stanford Heuristic Programming Project report HPP-83-12.
- [69] John Stuart Mill. *A System of Logic: The Principles of Evidence and the Methods of Scientific Investigation*. Harper and Brothers, 1900.
- [70] G.F. Miozzari and C. Yanofsky. The regulatory region of the trp operon of *S. marcescens*. *Journal of Bacteriology*, 133(3):1457-1466, 1978.
- [71] G.F. Miozzari and C. Yanofsky. Translation of the leader region of the *E. coli* tryptophan operon. *Nature*, 276(5689):684-689, 1978.
- [72] P.H. Morris and R.A. Nado. Representing actions with an assumption-based truth maintenance system. In *Proceedings of the 1986 National Conference on Artificial Intelligence*, pages 13-17. Morgan Kaufmann Publishers, 1986.
- [73] D.E. Morse and A.N.C. Morse. Dual-control of the tryptophan operon is mediated by both Tryptophanyl-tRNA synthetase and the repressor. *Journal of Molecular Biology*, 103:209-226, 1976.
- [74] A. Newell and H. A. Simon. GPS, a program that simulates human thought. In *Computers and Thought*, pages 279-293. McGraw-Hill, 1963.

- [75] A. Newell and H.A. Simon. *Human Problem Solving*. Prentice Hall, 1972.
- [76] B. Nichols, M. van Cleemput, and C. Yanofsky. Nucleotide sequence of *E. coli trpE* anthranilate synthetase component I contains no tryptophan residues. *Journal of Molecular Biology*, 146:45-54, 1981.
- [77] N.J. Nilsson. *Principles of Artificial Intelligence*. Morgan Kaufmann, Los Altos, CA, 1980.
- [78] D.L. Oxender, G. Zurawski, and C. Yanofsky. Attenuation in the *E. coli* tryptophan operon: Role of RNA secondary structure involving the tryptophan codon region. *Proceedings of the National Academy of Sciences, USA*, 76(11):5524-5528, 1979.
- [79] E.P.D. Pednault. Extending conventional planning techniques to handle actions with context-dependent effects. In *Proceedings of the 1988 National Conference on Artificial Intelligence*, pages 55-59. Morgan Kaufmann Publishers, 1988.
- [80] T. Platt. Termination of transcription and its regulation in the tryptophan operon of *E. Coli*. *Cell*, 24:10-23, 1981.
- [81] T. Platt, C. Squires, and C. Yanofsky. Ribosome-protected regions in the leader-*trpE* sequence of *E. coli* tryptophan operon messenger RNA. *Journal of Molecular Biology*, 103:411-420, 1976.
- [82] T. Platt and C. Yanofsky. An intercistronic region and ribosome-binding site in bacterial messenger RNA. *Proceedings of the National Academy of Sciences, USA*, 72(6):2399-2403, 1975.
- [83] K.R. Popper. *Conjectures and Refutations: The Growth of Scientific Knowledge*. Harper, 1965.

- [84] S.A. Rajamoney. *Explanation-Based Theory Revision: An Approach to the Problems of Incomplete and Incorrect Theories*. PhD thesis, University of Illinois Department of Computer Science, 1988.
- [85] C. Rich and H.E. Shrobe. Initial report on a LISP programmer's apprentice. Technical Report TR-354, Massachusetts Institute of Technology, 1976.
- [86] D. Rose and P. Langley. Chemical discovery as belief revision. *Machine Learning*, 1:423-451, 1986.
- [87] J.K. Rose and C. Yanofsky. Metabolic regulation of the tryptophan operon of *E. coli*: Repressor-independent regulation of transcription initiation frequency. *Journal of Molecular Biology*, 69:103-118, 1972.
- [88] A.D. Round. QSOPS: a workbench environment for the qualitative simulation of physical processes. Technical Report KSL-87-37, Stanford University Knowledge Systems Laboratory, 1987. Also appears in Proceedings of the European Simulation Multiconference.
- [89] E. Sacks. Qualitative mathematical reasoning. Technical Report MIT/LCS/TR-329, Massachusetts Institute of Technology Laboratory for Computer Science, 1985.
- [90] E. Sacks. Qualitative mathematical reasoning. In *Proceedings of the 1985 International Joint Conference on Artificial Intelligence*, pages 137-139. Morgan Kaufmann Publishers, 1985.
- [91] R. Simmons. Commonsense arithmetic reasoning. In *Proceedings of the 1986 National Conference on Artificial Intelligence*, pages 118-124. Morgan Kaufmann Publishers, 1986.
- [92] R. Simmons and J. Mohammed. Causal modeling of semiconductor fabrication. Technical Report 65, Schlumberger Palo Alto Research, 1987.

- [93] R.G. Simmons. *Combining Associational and Causal Reasoning to Solve Interpretation and Planning Problems*. PhD thesis, Massachusetts Institute of Technology, September 1988.
- [94] H.A. Simon. *The Sciences of the Artificial*. MIT Press, 1969.
- [95] H.A. Simon. Why should machines learn? In *Machine Learning*. Tioga, 1982.
- [96] V. Soo. *A Qualitative Matching Scheme For Postulating Enzyme Kinetic Models and Experimental Conditions: Reasoning With Constraints*. PhD thesis, Rutgers University, 1987.
- [97] C. Squires, F. Lee, K. Bertrand, C.L. Squires, M.J. Bronson, and C. Yanofsky. Nucleotide sequence of the 5' end of tryptophan messenger RNA of *E. coli*. *Journal of Molecular Biology*, 103:351-381, 1976.
- [98] C.L. Squires, F.D. Lee, and C. Yanofsky. Interaction of the trp repressor and RNA polymerase with the trp operon. *Journal of Molecular Biology*, 92:93-111, 1975.
- [99] M. Stefik and D.G. Bobrow. Object-oriented programming: Themes and variations. *AI Magazine*, 6(4):40-62, 1986.
- [100] M.J. Stefik. *Planning with Constraints*. PhD thesis, Stanford University Computer Science Department, January 1980. See also Stanford University Computer Science Department reports HPP-80-2, STAN-CS 80-784.
- [101] I. Stroynowski and C. Yanofsky. Transcript secondary structures regulate transcription termination at the attenuator of *S. marcescens* tryptophan operon. *Nature*, 298(5869):34-38, 1982.
- [102] G.J. Sussman. *A computational model of skill acquisition*. PhD thesis, Massachusetts Institute of Technology AI Laboratory, 1973.

- [103] P. Thagard and K. Holyoak. Discovering the wave theory of sound: Inductive inference in the context of problem solving. In *Proceedings of the 1985 International Joint Conference on Artificial Intelligence*, pages 610–612. Morgan Kaufmann Publishers, 1985.
- [104] C.H. Tong. *Knowledge-Based Circuit Design*. PhD thesis, Stanford University Computer Science Department, 1988.
- [105] D.S. Weld. The use of aggregation in causal simulation. *Artificial Intelligence*, 30:1–34, 1986.
- [106] D.C. Wilkins. *Apprenticeship Learning Techniques for Knowledge Based Systems*. PhD thesis, Stanford University, 1988. See also Stanford Computer Science Department reports STAN-CS-88-1242, KSL-88-14.
- [107] B.C. Williams. Doing time: Putting qualitative reasoning on firmer ground. In *Proceedings of the 1986 National Conference on Artificial Intelligence*, pages 105–112. Morgan Kaufmann Publishers, 1986.
- [108] P.H. Winston. *Artificial Intelligence*. Addison-Wesley, 1984.
- [109] C. Yanofsky. Attenuation in the control of expression of bacterial operons. *Nature*, 289:751–758, 1981.
- [110] C. Yanofsky. Personal communication. 1987.
- [111] C. Yanofsky. Personal communication. 1989.
- [112] C. Yanofsky, T. Platt, I.P. Crawford, B.P. Nichols, G.E. Christie, H. Horowitz, M. van Cleemput, and A.M. Wu. The complete nucleotide sequence of the tryptophan operon of *E. coli*. *Nucleic Acids Research*, 9(24):6647–6668, 1981.
- [113] C. Yanofsky and L. Soll. Mutations affecting trp-tRNA and its charging and their effect on regulation of transcription termination at the attenuator of the tryptophan operon.

Journal of Molecular Biology, 113:663-677, 1977.

- [114] G. Zurawski, K. Brown, D. Killingly, and C. Yanofsky. Nucleotide sequence of the leader region of the phenylalanine operon of *E. coli*. *Proceedings of the National Academy of Sciences, USA*, 75(10):4271-4275, 1978.
- [115] G. Zurawski and C. Yanofsky. *E. coli* tryptophan operon leader mutations, which relieve transcription termination, are *cis*-dominant to *trp* leader mutations, which increase transcription termination. *Journal of Molecular Biology*, 142:123-129, 1980.